ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αναλογική καθοδήγηση της ανάλυσης και προδιαγραφής λογισμικού

Μιράντα Μ.Β. Κυπριώτη

Μεταπτυχιακή Εργασία

Ηράκλειο, 1996

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αναλογική καθοδήγηση της ανάλυσης και προδιαγραφής λογισμικού

Εργασία που υποβλήθηκε από την
Μιράντα Μ.Β. Κυπριώτη
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Μιράντα Μ.Β. Κυπριώτη
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Πάνος Κωνσταντόπουλος
Αναπληρωτής Καθηγητής, Επόπτης

Γιώργος Γεωργακόπουλος
Επίκουρος Καθηγητής, Μέλος

Απόστολος Τραγανίτης
Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος
Αναπληρωτής Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, 1996

Στη Μιράντα Α. Σαββάκη

Αναλογική καθοδήγηση της ανάλυσης και προδιαγραφής λογισμικού

Μιράντα Κυπριώτη
Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Μέχρι πρόσφατα, η δημιουργία ενός συστήματος λογισμικού ήταν παραδοσιακά προσανατολισμένη στο προϊόν που κατασκευάζεται. Τελευταία οι μηχανικοί λογισμικού έχουν αρχίσει να αναγνωρίζουν το γεγονός ότι η κατασκευή ποιοτικών προϊόντων εξαρτάται σε μεγάλο βαθμό από τη διαδικασία με την οποία παράγονται. Για τη διαχείριση και τον έλεγχο της διαδικασίας αυτής αναπτύχθηκαν και χρησιμοποιούνται μοντέλα διαδικασιών. Βασικός στόχος της κατασκευής μοντέλων διαδικασιών είναι η καθοδήγηση των ανθρώπων που σχετίζονται με τη διαδικασία. Σαν καθοδήγηση, αναφέρεται γενικά η δυνατότητα παροχής συστάσεων και υλικού αναφοράς για τη διευκόλυνση της εκτέλεσης της διαδικασίας από ανθρώπους.

Εστιάζουμε την προσοχή μας εκεί όπου θεωρούμε ότι υπάρχει ουσιαστική ανάγκη για καθοδήγηση, δηλαδή σε περιπτώσεις όπου η μεθοδολογία που περιγράφεται από το μοντέλο προτείνει διάφορες εναλλακτικές λύσεις και δεν είναι προφανές ποια είναι η περισσότερο αρμόζουσα. Οι αναλογίες που εντοπίζονται με ανάλυση ομοιότητας μεταξύ περιπτώσεων εκτέλεσης μιας διαδικασίας, μπορούν να παρουσιάσουν συγκεκριμένα παραδείγματα για το πώς πότε και γιατί πρέπει να ακολουθηθούν συγκεκριμένα βήματα σε μια διαδικασία.

Στην εργασία αυτή επιχειρήθηκε με κατάλληλη παράσταση της διαδικασίας και των ιχνών να επιλεγούν και να παρουσιαστούν τα κατάλληλα, δηλαδή συγκεκριμένα και σχετικά, παραδείγματα που θα διευκολύνουν τη λήψη απόφασης σε κατάσταση διλήμματος. Για την επίτευξη του σκοπού αυτού χρησιμοποιήθηκε ένα αποφασεοκεντρικό μοντέλο για διαδικασίες, που το αποτέλεσε το μοντέλο πληροφορίας μας στην Telos και ένα γενικό μοντέλο αναλογικής ομοιότητας σημασιολογικών περιγραφών για τον υπολογισμό της αναλογίας με τα αποθηκευμένα παραδείγματα-ίχνη της διαδικασίας.

Τελικά , η προσέγγισή μας είναι εφαρμόσιμη σε μεγάλο μέρος της διαδικασίας παραγωγής προδιαγραφών όπου υπάρχουν τυπικές περιγραφές του προϊόντος.Η αποτελεσματικότητά της όμως, εξαρτάται από το πλήθος και την ποιότητα

των ιχνών που υπάρχουν στη βάση, γι'αυτό στο τέλος διατυπώνονται κάποιες προτάσεις βελτίωσης.

Επόπτης: Πάνος Κωνσταντόπουλος,
Αναπληρωτής καθηγητής,
Πανεπιστήμιο Κρήτης.

Analogical guidance of software analysis and specification process

Miranta Kiprioti
Master's Thesis

Computer Science Department
University of Crete, Greece

Abstract

Traditionally , software systems development focused on the product part of the system.Recently, the software engineering community realized that the quality of software products depends highly on the software production process. Process models have been developed to manage and control the software production process.

Among the goals of process modeling is the guidance of humans who are related to the process.Guidance refers to the provision of reference material and suggestions to facilitate human enactment of the process.

This work focuses on where we think that there is a real need for guidance. Specifically , the focus is in cases where the software engineer who follows a methodology is faced with a set of alternatives all of which are applicable. Similarity analysis between traced enactments of processes could aid new enactments of these models.Similarity analysis detects analogies between traced enactments which reveal concrete examples on how, when and why certain steps in a process should be followed.

This work aimed at selecting and presenting the appropriate (which means concrete and relevant) examples. In order to accomplish this we make use of a decision oriented process model , which practically is our information model in Telos and a general model of analogical similarity of conceptual descriptions for the computation of similarity between the examples-traces of the process.

At the end, we conclude that our approach is applicable through a large part of the requirements specification process, when formal descriptions of the project do exist. However , the effectiveness of our approach depends highly on the amount and quality of the traces that exist in the repository. Therefore , se suggest ways of improvement to overcome this problem.

Advisor: Panos Constantopoulos
Associate Professor,
University of Crete.

Ευχαριστίες

Στο σημείο αυτό θά θελα να πω ένα μεγάλο ευχαριστώ σε όλους όσους με τον ένα ή τον άλλο τρόπο με στήριξαν κατά τη διάρκεια της προσπάθειας για την πραγματοποίηση αυτής της εργασίας.

Πρώτα απ' όλα ευχαριστώ το Τμήμα Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης που με έκανε δεκτή στο μεταπτυχιακό του πρόγραμμα καθώς και το Ινστιτούτο Πληροφορικής του Ι.Τ.Ε. για την οικονομική και υλικοτεχνική υποστήριξη που μου παρείχε.

Ειδικότερα, οφείλω ένα μεγάλο ευχαριστώ στον επόπτη μου, τον Πάνο Κωνσταντόπουλο, για τις υποδείξεις και τα σχόλιά του. Τον ευχαριστώ γιατί ως πραγματικός Δάσκαλος, προσπάθησε να ενισχύσει την αποφασιστικότητα και την αποτελεσματικότητά μου δείχνοντάς μου εμπιστοσύνη και προτροπές χωρίς διάθεση να επιβάλει το δικό του τρόπο σκέψης.

Σημαντική για την εργασία αυτή ήταν η συμβολή του Γιώργου του Σπανουδάκη που με τις υποδείξεις και τις ιδέες του με βοήθησε στα αρχικά στάδια της εργασίας. Αλλά και στη συνέχεια, προσέφερε πρόθυμα τις παρατηρήσεις και τις συμβουλές του κάθε φορά που του το ζητούσα. Μέσα από τις συζητήσεις μας για τη χρησιμότητα της αναλογικής ομοιότητας, άρχισε να παίρνει μορφή η ιδέα της αναλογικής καθοδήγησης.

Θέλω ακόμη να ευχαριστήσω τους γονείς μου, Μιχάλη και Βαρβάρα, για την πίστη τους σε μένα, το κουράγιο και την οικονομική και ηθική στήριξη που μου πρόσφεραν όλα αυτά τα χρόνια. Χωρίς τη συμπαράστασή τους, αυτή η εργασία δύσκολα θα είχε ολοκληρωθεί.

Σημαντική συμπαράσταση είχα και απο πολλούς φίλους, συνάδελφους μεταπτυχιακούς φοιτητές, που παρά το δικό τους φόρτο εργασίας με υπομονή και όρεξη ήταν πάντα διαθέσιμοι για σχόλια και παρατηρήσεις σε πολλά στάδια της εργασίας, αλλά και για ψυχολογική υποστήριξη στις δύσκολες ώρες της τελικής ευθείας. Τους ευχαριστώ θερμά γι'αυτό.

Θα ήταν παράλειψη να μην ευχαριστήσω εδώ τον Τάσο το Σταματούρο για τη συμβολή του στο "ξεκαθάρισμα" πολλών εννοιών που με υπομονή κι επιμονή ζητούσε να του εξηγήσω, αλλά και για την κατανόηση που έδειχνε στις άπειρες ώρες δουλειάς που απαιτούσε η εργασία αυτή, και που στάθηκε ψύχραιμος δέκτης της όλης φόρτισης που κουβαλούσα.

Ακόμη θέλω να ευχαριστήσω και τα μέλη της Ομάδας Πληροφοριακών Συστημάτων και Τεχνολογίας Λογισμικου για τη σημαντική βοήθειά τους στα προβλήματα της υλοποίησης και για τις ενδιαφέρουσες συζητήσεις που είχαμε κατά καιρούς.

Περιεχόμενα

Περίληψη **i**

Abstract **iv**

Ευχαριστίες **vi**

Περιεχόμενα **1**

1 Εισαγωγή **3**
1.1 Το πρόβλημα 3
1.2 Η προσέγγισή μας 4
1.3 Συνολική άποψη του κειμένου 6

2 Μοντέλα διαδικασιών: επισκόπηση τεχνολογίας αιχμής **7**
2.1 Μοντέλα διαδικασιών 7
2.2 Προσεγγίσεις στην κατασκευή μοντέλων 8
2.2.1 Σκοποί κατασκευής των μοντέλων διαδικασιών 8
2.2.2 Βασικές Έννοιες που χρησιμοποιούνται 9
2.2.3 Γλώσσες παράστασης διαδικασιών 10
2.3 Κατηγορίες μοντέλων 11
2.4 Συστήματα 13
2.4.1 Εκτέλεση της διαδικασίας και καθοδήγηση 19
2.5 IPSW 6 20

3 Οντοκεντρικές Μέθοδοι Σχεδίασης και Ανάλυσης **21**
3.1 Συνοπτική θεώρηση των μεθόδων OOA- OOD 21

4 Ανάλυση ομοιότητας **25**
4.1 Γλώσσα παράστασης γνώσης Telos 25
4.2 Αναλογική ομοιότητα περιγραφών 26
4.3 Αναλογική ομοιότητα στα πλαίσια μοντέλων διαδικασιών 28

4.4 Παράδειγμα εντοπισμού αναλογιών 29

5 Καθοδήγηση της διαδικασίας παραγωγής προδιαγραφών: Ένα παράδειγμα 33

5.1 Το μοντέλο της διαδικασίας 33
5.2 Το πρόβλημα 35
5.3 Οι αποθηκευμένες περιπτώσεις (traces) 36
5.4 Η λύση 37
5.5 Μέθοδος 39
5.6 Συμπεράσματα 39

6 Η προσέγγισή μας 41

6.1 Το μεταμοντέλο 42
6.2 Το μοντέλο της διαδικασίας BOOD 45
6.2.1 Περιστάσεις [περιβαλλόντων απόφασης](Situations) 45
6.2.2 Ανάλυση περιβαλλόντων απόφασης 48
6.2.3 Το μοντέλο του προϊόντος (Product Model) 56
6.2.4 Παρατηρήσεις 64
6.3 Καθοδήγηση μέσω ομοιότητας κατά την εκτέλεση του μοντέλου . 66
6.3.1 Προσομοίωση της ”εκτέλεσης” του μοντέλου 66
6.3.2 Υποστήριξη χρήσης αναλογικού συλλογισμού 68
6.3.3 Παραγωγή ιχνών 69

7 Επίλογος 73

7.1 Εφαρμοσιμότητα 73
7.2 Αποτίμηση 74
7.3 Αδυναμίες- επεκτάσεις- μελλοντικές κατευθύνσεις 75

Βιβλιογραφία 77

Α Αγγλοελληνικό Ευρετήριο Όρων 81

Β Ελληνοαγγλικό Ευρετήριο Όρων 85

Γ Ορισμοί 89

Δ Περιγραφές σε Telos 93

Κεφάλαιο 1

Εισαγωγή

Στην εργασία αυτή παρουσιάζουμε έναν τρόπο καθοδήγησης της διαδικασίας διατύπωσης των απαιτήσεων ενός πληροφοριακού συστήματος, στη φάση που χειριζόμαστε ημιτυπικές ή/και τυπικές περιγραφές του υποσυστήματος λογισμικού.

Μέχρι πρόσφατα, η δημιουργία ενός συστήματος λογισμικού ήταν παραδοσιακά προσανατολισμένη στο προϊόν που κατασκευάζεται. Τελευταία οι μηχανικοί λογισμικού έχουν αρχίσει να αναγνωρίζουν το γεγονός ότι η κατασκευή ποιοτικών προϊόντων εξαρτάται σε μεγάλο βαθμό από τη διαδικασία με την οποία παράγονται. Για τη διαχείριση και τον έλεγχο της διαδικασίας αυτής αναπτύχθηκαν και χρησιμοποιούνται μοντέλα διαδικασιών.

Βασικός στόχος της κατασκευής μοντέλων διαδικασιών είναι η καθοδήγηση των ανθρώπων που σχετίζονται με τη διαδικασία. Σαν καθοδήγηση, αναφέρεται γενικά η δυνατότητα παροχής συστάσεων και υλικού αναφοράς για τη διευκόλυνση της εκτέλεσης της διαδικασίας από ανθρώπους [CKO92]. Ο όρος καθοδήγηση χρησιμοποιείται για να εκφράσει την παροχή ενός μηχανισμού ερμηνείας των μοντέλων (Software Process Models), και μέσω αυτού την επιβολή περιορισμών των μοντέλων διαδικασιών [AFN94], με τη μορφή συνταγών ή απαγορεύσεων (prescriptions / proscriptions). Σ' αυτό το σημείο πρέπει να τονίσουμε ότι η ερμηνεία των μοντέλων δεν είναι απαραίτητα πλήρως αυτοματοποιημένη.

1.1 Το πρόβλημα

Το πρόβλημα που αντιμετωπίζουμε είναι το πώς να παρέχουμε καθοδήγηση στον μηχανικό λογισμικού που επιχειρεί να κατασκευάσει το μοντέλο τού συστήματός του ακολουθώντας κάποια μεθοδολογία. Στη διαδικασία παραγωγής τυπικών περιγραφών ο παράγοντας της ανθρώπινης δημιουργικότητας

είναι μέγιστης σημασίας, και γι'αυτό γενικότερα στην παραγωγή εννοιολογικών μοντέλων υπάρχει μια τάση να αποφεύγονται οι "συνταγές". Παρόλα' αυτά, υπάρχουν γνωστές και ευρέως διαδεδομένες μεθοδολογίες που δίνουν γενικές κατευθύνσεις για την παραγωγή π.χ. οντοκεντρικών προϊόντων λογισμικού. Προκειμένου να αντλήσει κανείς καθοδήγηση από τέτοιες γενικές μεθοδολογίες (που συνήθως περιγράφονται με αφηρημένους κανόνες) είναι χρήσιμο να μπορεί να δει συγκεκριμένα (concrete) σχετικά παραδείγματα τη στιγμή που βρίσκεται σε κάποιο μεθοδολογικό δίλημμα. Σ'αυτές τις περιπτώσεις η μεθοδολογία συνήθως αναφέρει απλά, ότι πρέπει να επιλεγεί "αυτό που ταιριάζει καλύτερα στην εφαρμογή".

Αυτό που προσπαθούμε να πετύχουμε σ'αυτή την εργασία, είναι η επιλογή και παρουσίαση των κατάλληλων, δηλαδή συγκεκριμένων και σχετικών, παραδειγμάτων. Εστιάζουμε την προσοχή μας εκεί όπου θεωρούμε ότι υπάρχει ουσιαστική ανάγκη για καθοδήγηση, δηλαδή σε περιπτώσεις που η μεθοδολογία προτείνει διάφορες εναλλακτικές λύσεις και δεν είναι προφανές ποια είναι η περισσότερο αρμόζουσα.

1.2 Η προσέγγισή μας

Ο τρόπος με τον οποίο προσεγγίζουμε το πρόβλημα συνίσταται καταρχήν στην γνωσιοκεντρική παράσταση του μοντέλου της διαδικασίας (σε Telos) και, στη συνέχεια, στη χρήση ενός μηχανισμού υπολογισμού αναλογικής ομοιότητας για την ανάκληση των κατάλληλων παραδειγμάτων. Συνδυάζοντας αυτά τα δύο, παρουσιάζουμε μια μέθοδο καθοδήγησης. Η προτεινόμενη μέθοδος βασίζεται αφ' ενός σε ένα **γενικό μοντέλο αναλογικής ομοιότητας σημασιολογικών περιγραφών**, το οποίο ορίσθηκε στη διδακτορική διατριβή του Γιώργου Σπανουδάκη [G. 94], και αφετέρου σε ένα **μοντέλο παράστασης μοντέλων διαδικασιών** που βασίζεται σ'ένα αποφασεοκεντρικό μεταμοντέλο περιγραφής διαδικασιών [GR94, ROL94].

Κάθε συγκεκριμένη εφαρμογή μιας μεθοδολογίας, στην οποία θα αναφερόμαστε και σαν "εκτέλεση του μοντέλου της διαδικασίας" θεωρούμε ότι μπορεί να καταγραφεί σαν ίχνος και να χρησιμοποιηθεί σαν παράδειγμα.

Χρησιμοποιούμε το μεταμοντέλο διαδικασιών για να περιγράψουμε τα μοντέλα διαδικασιών μας και τις περιπτώσεις εκτελέσεών τους, και το μοντέλο αναλογικής ομοιότητας για να επιλέξουμε τα παραδείγματα ακολούθησης του μοντέλου και να τα αξιολογήσουμε. Πιστεύουμε πως η ανάλυση ομοιότητας των ιχνών από εκτελέσεις μοντέλων διαδικασιών μπορεί να βοηθήσει επόμενες ενεργοποιήσεις τους. Οι αναλογίες που εντοπίζονται μπορούν να παρουσιά-

σουν συγκεκριμένα παραδείγματα για το πώς, πότε και γιατί θα πρέπει να ακολουθηθούν συγκεκριμένα βήματα σε μια διαδικασία.

Το μεταμοντέλο για τις διαδικασίες είναι ένα μοντέλο περιγραφής μεθόδων, που κι αυτές αποτελούν μοντέλα διαδικασιών. Έχει υποστηριχτεί ότι για την περιγραφή μιας διαδικασίας παραγωγής λογισμικού απαιτούνται τρία επίπεδα αφαίρεσης (βλ. σελ 11)· το μεταμοντέλο χρησιμεύει σαν το πιο αφηρημένο από αυτά, παρέχοντας τις βασικές έννοιες με τις οποίες περιγράφουμε μεθόδους.

Το συγκεκριμένο μεταμοντέλο που έχουμε επιλέξει είναι κατάλληλο για την περιγραφή μεθόδων σχεδίασης λογισμικού, γιατί πετυχαίνει να εκφράσει το γεγονός ότι σ'αυτές τις διαδικασίες προχωράμε παίρνοντας αποφάσεις πάνω σε κάποια ζητήματα και όχι ακολουθώντας μια προδιαγεγραμμένη σειρά βημάτων.

Οι βασικές έννοιες που εισάγονται από το μεταμοντέλο διαδικασιών, είναι αρκετές για να περιγράψουμε τις ακολουθίες των αποφάσεων που οδηγούν τη διαδικασία. Συγκεκριμένα, βασική είναι η έννοια του **περιβάλλοντος απόφασης** (context) το οποίο περιγράφεται (εξαρτάται) από την **περίσταση** δηλαδή την κατάσταση στην οποία βρίσκεται το εν εξελίξει προϊόν και την **απόφαση** που παίρνει ο σχεδιαστής. Η απόφαση αυτή μπορεί να έχει και κάποια **επιχειρήματα** που σχετίζονται μ'αυτήν (υπέρ ή κατά). Το περιβάλλον απόφασης, μπορεί να είναι **εκτελέσιμο** οπότε αλλάζει άμεσα το **προϊόν** και έμμεσα την περίσταση, ή να περιγράφει ένα **σημείο επιλογής μεταξύ εναλλακτικών βημάτων** ή ακόμη να περιγράφει μια **ακολουθία βημάτων**, που πρέπει να εκτελεστούν όλα προκειμένου να πραγματοποιηθεί η απόφαση. Τα βήματα αυτά είναι επίσης περιβάλλοντα απόφασης, κι έτσι το μοντέλο μας δίνει τη δυνατότητα να σχηματίζουμε ιεραρχίες από περιβάλλοντα απόφασης όταν περιγράφουμε διαδικασίες.

Το μοντέλο αναλογικής ομοιότητας που αναφέρθηκε, εντοπίζει αναλογίες μεταξύ αντικειμένων, και εκτιμά την ομοιότητά τους με βάση τις σημασιολογικές τους περιγραφές. Οι περιγραφές αυτές χρησιμοποιούν τρεις βασικούς γενικούς μηχανισμούς αφαίρεσης: γενίκευση, ταξινόμηση και απόδοση γνωρισμάτων. Η ομοιότητα υπολογίζεται συνδυάζοντας τις μερικές αποστάσεις: γενίκευσης, ταξινόμησης και απόδοσης γνωρισμάτων. Αυτές οι αποστάσεις εκτιμούνται λαμβάνοντας υπόψιν ιδιότητες των γνωρισμάτων όπως χαρακτηριστικότητα (*characticity*), βαθμός αφαίρεσης (*abstractness*) και προσδιοριστικότητα (*determinance*), που χαρακτηρίζουν τη σπουδαιότητα (*salience*) των γνωρισμάτων στον υπολογισμό των αναλογιών. Αυτό το μοντέλο ομοιότητας είναι ανεξάρτητο του πεδίου εφαρμογής καθώς και του τρόπου παράστασης των αντικειμένων.

Το μοντέλο με το οποίο παριστάνουμε την διαδικασία που πρόκειται να ακο-

λουθηθεί, και εκείνο με το οποίο παριστάνουμε τις περιπτώσεις των εκτελέσεών της έχουν κοινές συνιστώσες (παράγονται από το ίδιο μεταμοντέλο) πράγμα που διευκολύνει τον εντοπισμό αναλογιών μεταξύ τους. Επιπλέον ο τρόπος που έχουμε επιλέξει να περιγράψουμε σε Telos τις συνιστώσες του μεταμοντέλου διαδικασιών, ικανοποιεί το στόχο της ανάδειξης των ανάλογων συστατικών των καταστάσεων, που, όπως αναφέρθηκε, παίζουν σημαντικό ρόλο στην καθοδήγηση όπως εμείς την εννοούμε.

1.3 Συνολική άποψη του κειμένου

Μετά τη συνοπτική περιγραφή της εργασίας αυτής, ακολουθεί στο κεφάλαιο 2, μια επισκόπηση της τεχνολογίας αιχμής στον τομέα των μοντέλων διαδικασιών.

Στο κεφάλαιο 3 θα δούμε συνοπτικά τις πιο γνωστές μεθόδους ανάλυσης και σχεδίασης που έχουν οντοκεντρική φιλοσοφία.

Στο κεφάλαιο 4 θα περιγραφεί το μοντέλο αναλογικής ομοιότητας, και η συγκεκριμένη εφαρμογή του στη γλώσσα Telos.

Στο κεφάλαιο 5 θα δούμε με τη βοήθεια ενός μικρού παραδείγματος, που είναι εύκολο στην κατανόηση, πώς εννοούμε την μέθοδο καθοδήγησης που προτείνουμε στα πλαίσια της εργασίας μας.

Στο κεφάλαιο 6, που αποτελεί και τον πυρήνα της εργασίας μας, περιγράφουμε αναλυτικά την προσέγγισή μας, εισάγοντας αρχικά το μεταμοντέλο και στη συνέχεια εκφράζοντας τη μεθοδολογία οντοκεντρικής ανάλυσης και σχεδίασης του Booch σαν περίπτωση αυτού του μεταμοντέλου, και τελειώνοντας με την περιγραφή της καθοδήγησης αυτής της συγκεκριμένης μεθόδου.

Στο κεφάλαιο 7 συνοψίζουμε τα συμπεράσματά μας από την εργασία αυτή, και εκτιμούμε τα δυνατά και αδύνατα σημεία της, και περιγράφουμε τις μελλοντικές κατευθύνσεις εξέλιξης και επέκτασης του μηχανισμού της καθοδήγησης που προτείναμε.

Στο παράρτημα Α δίνεται ένα αγγλοελληνικό ευρετήριο των κυριότερων όρων που χρησιμοποιήθηκαν στο κείμενο της γραπτής εργασίας και στο παράρτημα Β το αντίστοιχο ελληνοαγγλικό.

Στο παράρτημα Γ υπάρχουν ορισμοί εννοιών από τη βιβλιογραφία.

Στο παράρτημα Δ δίνονται οι περιγραφές σε Telos του μεταμοντέλου και του μοντέλου της διαδικασίας Booch καθώς και κάποιων περιπτώσεων εκτέλεσής της.

Τέλος, ακολουθεί ένας κατάλογος των βιβλιογραφικών αναφορών που εμφανίζονται στη γραπτή εργασία.

Κεφάλαιο 2

Μοντέλα διαδικασιών: επισκόπηση τεχνολογίας αιχμής

Μοντέλα διαδικασιών συναντάμε σε διαφορετικά πεδία εφαρμογής. Από τη μια μπορούμε να διακρίνουμε τις Business processes (Enterprise Modeling, Process Reengineering, Computer Integrated Manufacturing, Coordination Technology) όπου μοντελοποιείται ολόκληρη η διαδικασία παραγωγής μιας επιχείρησης, ενώ από την άλλη τα μοντέλα της διαδικασίας ανάπτυξης λογισμικού (Information Systems Engineering, Workflow management).

Σ'αυτό το κεφάλαιο επιχειρείται μια επισκόπηση του ερευνητικού πεδίου που αφορά μοντέλα διαδικασιών στο χώρο της παραγωγής λογισμικού. Στόχος μας εδώ είναι να παρουσιάσουμε τις έννοιες που συναντά κανείς σ'αυτό το χώρο για να ξεκαθαρίσουμε θέματα σχετικά με την ορολογία.

Γι'αυτό, γίνεται μια συνολική παρουσίαση των ερευνητικών προγραμμάτων και συστημάτων που υποστηρίζουν μοντέλα διαδικασιών εκθέτοντας την προσέγγιση που ακολουθούν στη μοντελοποίηση (σκοπός του μοντέλου, βασικές έννοιες και τρόπος περιγραφής τους), τις γλώσσες που χρησιμοποιούν για την περιγραφή των μοντέλων, και τους μηχανισμούς που διαθέτουν για την εκτέλεση (enactment) των διαδικασιών.

2.1 Μοντέλα διαδικασιών

Με τον όρο **μοντέλο διαδικασίας** (process model) αναφερόμαστε σε μια σαφή κατασκευή που παριστάνει μια αφαίρεση κάποιας διαδικασίας του πραγματικού κόσμου.

Με τον όρο **διαδικασία** (process) αναφερόμαστε σε ένα σύνολο από μερικώς διατεταγμένες δραστηριότητες οι οποίες στοχεύουν στην επίτευξη κάποιου

στόχου και στους σχετιζόμενους με τις δραστηριότητες αυτές πόρους (άνθρωποι - εργαλεία - χρόνος) και προϊόντα.

Διαδικασία παραγωγής λογισμικού (software production process ή απλά software process) λέμε το σύνολο των δραστηριοτήτων, κανόνων, τεχνικών και εργαλείων που χρησιμοποιούνται στις επιχειρήσεις παραγωγής λογισμικού.

Ένα **μοντέλο διαδικασίας παραγωγής λογισμικού** (software process model) λοιπόν, είναι μια σαφής κατασκευή που παριστάνει μια αφαίρεση για τη διαδικασία παραγωγής λογισμικού. Αυτό θα εννοούμε στη συνέχεια με τον όρο **μοντέλο διαδικασίας**, για συντομία(ΜΔ).

Όταν μιλάμε για **enactment** (εκτέλεση, εφαρμογή του μοντέλου), μιας διαδικασίας, εννοούμε ότι το μοντέλο πραγματοποιείται όπως έχει, και δεν περιλαμβάνεται καμμιά έννοια μετάφρασης του μοντέλου σε λογισμικό, (θεωρούμε ότι κάποια τμήματα εκτελούνται από υπολογιστή και άλλα από ανθρώπους).

2.2 Προσεγγίσεις στην κατασκευή μοντέλων

Έχουν εμφανιστεί στη βιβλιογραφία πολλά μοντέλα διαδικασιών. Για να δούμε συνοπτικά τις διάφορες προσεγγίσεις θα εξετάσουμε :
α. τον στόχο που έχει η κατασκευή του μοντέλου,
β. τις βασικές έννοιες που περιγράφει το μοντέλο,
γ. τον τρόπο παράστασης των εννοιών αυτών.

2.2.1 Σκοποί κατασκευής των μοντέλων διαδικασιών

Συνοψίζοντας από τα [CKO92, KTO93, ABCM92, AFN94] οι αντικειμενικοί στόχοι των ΜΔ, οι βασικοί λόγοι δηλαδή για τους οποίους θεωρείται αναγκαία η ύπαρξη των ΜΔ είναι οι παρακάτω:

▷ διευκόλυνση της κατανόησης από ανθρώπους και της επικοινωνίας μεταξύ τους. Αυτό επιτυγχάνεται με την παράσταση της διαδικασίας σε μορφή κατανοητή από ανθρώπους. Επιπλέον, η τυποποίηση της διαδικασίας μέσω ενός μοντέλου θα πρέπει να συμβάλλει στη μεγαλύτερη αποδοτικότητα της ομαδικής δουλειάς, στην καλύτερη και σαφέστερη κατανόηση και συμφωνία ως προς το τί πρέπει να γίνει.

▷ υποστήριξη της βελτίωσης της διαδικασίας κάνοντας δυνατή την **αναχρησιμοποίηση**, σύγκριση, **ανάλυση**, **μέτρηση** και ελεγχόμενη εξέλιξη των διαδικασιών καθώς και την κατασκευή νέων.

▷ υποστήριξη της **διαχείρισης** της διαδικασίας όπου περιλαμβάνονται : έλεγχος, συντονισμός, πρόβλεψη και σχεδιασμός - προγραμματισμός (**planning**)

▷ αυτοματοποιημένη υποστήριξη της **εκτέλεσης** κάποιων τμημάτων της διαδικασίας. Επιπλέον, η αυτοματοποίηση της διαδικασίας θα πρέπει να υποστηρίζει τη συλλογική εργασία, την αυτόματη συλλογή μετρήσεων και δεδομένων που αντανακλούν την πραγματική εμπειρία για τη διαδικασία. Σ'αυτό το στόχο συμβάλλει και η δυνατότητα επιβολής περιορισμών (**enforcing**) ώστε να εξασφαλίζεται η ακεραιότητα της διαδικασίας.

▷ αυτοματοποίηση της **καθοδήγησης** για την εκτέλεση της διαδικασίας με τον ορισμό ενός αποδοτικού περιβάλλοντος ανάπτυξης λογισμικού, και την παροχή υποδείξεων και υλικού αναφοράς για τη διευκόλυνση της εκτέλεσης της διαδικασίας, καθώς και με τη διατήρηση αναχρησιμοποιήσιμων παραστάσεων της διαδικασίας σ'ένα repository.

Θα επιμείνουμε λίγο στην έννοια της καθοδήγησης κατά την εφαρμογή ενός ΜΔ. Πιο συγκεκριμένα, λοιπόν, με τον όρο **guidance** (καθοδήγηση) αναφερόμαστε στην παροχή ευκολιών όπως ([AFN94] σελ.154)

- Πληροφόρηση του χρήστη για τις δραστηριότητες που μπορούν να εκτελεσθούν από μια δεδομένη κατάσταση της εν εξελίξει διαδικασίας.
- Οδηγίες (για την ακρίβεια μια ακολουθία, ένα μονοπάτι από βήματα-δραστηριότητες) για την πραγματοποίηση κάποιου στόχου, όπως π.χ. κατασκευή ενός αντικειμένου με δεδομένο τύπο.
- βοήθεια στην εκτέλεση μιας εργασίας (π.χ. πώς δουλεύει κάτι)
- ανάλυση των συνεπειών μιας πράξης και εκτίμηση των συνεπειών ειδικά αν η πράξη δύναται να αλλάξει κάτι στο μοντέλο ή στα αποτελέσματα της εν εξελίξει διαδικασίας. (whatif analysis)
- αναφορά προς το χρήστη της τρέχουσας κατάστασης της διαδικασίας, και πώς έχει επιτευχθεί (ιστορία - τί έχει γίνει)
- παροχή εξηγήσεων σε περίπτωση που το μοντέλο απαγορεύει κάποια πρωτοβουλία του χρήστη.
- Μάθηση από το παρελθόν μέσω παρατηρήσεων (observations), για την παροχή ανατροφοδότησης (feedback) στο σύστημα και τη βοήθεια στους χρήστες να μάθουν να το χρησιμοποιούν.

2.2.2 Βασικές Έννοιες που χρησιμοποιούνται

Για την παραγωγή μοντέλων διαδικασιών οι συνηθέστερες και βασικότερες έννοιες, που χρησιμοποιούνται προκειμένου να περιγραφούν τα συστατικά μιας διαδικασίας, είναι οι παρακάτω:

activity : δραστηριότητα, ένα βήμα της διαδικασίας που προκαλεί εμφανείς αλλαγές κατάστασης στο προϊόν. Ενσωματώνει και υλοποιεί διαδικασίες προγραμματισμού, κανόνες και πολιτικές και στοχεύει στη δημιουργία ή τροποποίηση ενός προϊόντος.

artifact ή product : προϊόν, το κατασκεύασμα που προκύπτει σε κάποια στάδια της διαδικασίας αλλά και η πρώτη ύλη, αφού τα προϊόντα μιας διαδικασίας μπορούν να χρησιμοποιηθούν από κάποια άλλη για να μετασχηματιστούν. Το σύνολο των atrifacts που τελικά παραδίδονται προς χρήση το ονομάζουμε προϊόν λογισμικού (software product). Συχνά παρουσιάζεται η ανάγκη να υπάρχουν διαφορετικές εκδόσεις (versions) των προϊόντων.

agent ή actor : πράκτορας, υποκείμενο, αυτός που εκτελεί κάποια τμήματα της διαδικασίας.Ο όρος αυτός χρησιμοποιείται τόσο για ανθρώπους, όσο και για μηχανές.

role : ένα σύνολο από δικαιώματα και υπευθυνότητες, που ανατίθεται σε κάποιο agent προκειμένου να εκτελέσει κάποια εργασία.

tool : τα εργαλεία (προγράμματα υπολογιστών) που χρησιμοποιούνται για υποστηρίξουν ή να αυτοματοποιήσουν κάποια δραστηριότητα. Διαχωρίζονται σε active και passive ανάλογα με το αν μπορούν να ξεκινήσουν μια αλληλεπίδραση ή αν πρέπει να τα ξεκινήσει κάποιος άνθρωπος.

constraint :περιορισμός

Ειδικά σε κάποια περιβάλλοντα που υποστηρίζουν ταυτόχρονη επέμβαση από πολλούς χρήστες εμφανίζεται και η έννοια του **χώρου εργασίας (workspace)** της **δοσοληψίας** καθώς και της επικοινωνίας όπως στα ALF, ADELE, EPOS.

2.2.3 Γλώσσες παράστασης διαδικασιών

Γλώσσες παράστασης διαδικασιών ονομάζουμε τους διάφορους φορμαλισμούς που χρησιμοποιούνται για να περιγράψουν (παραστήσουν) τις παραπάνω βασικές έννοιες και σχέσεις μεταξύ τους. Παρουσιάζουμε εν συντομία παρακάτω,τους διάφορους τρόπους παράστασης δραστηριοτήτων που χρησιμοποιούνται από τέτοιες γλώσσες. Υπενθυμίζουμε εδώ ότι η διαδικασία είναι ένα σύνολο από μερικώς διατεταγμένες δραστηριότητες.

Κανόνες παραγωγής όπως στο MARVEL [BsKH92] όπου κάθε βήμα της διαδικασίας περικλείεται σε έναν κανόνα που έχει παραμέτρους και όνομα και αποτελείτα από μια συνθήκη, μια προαιρετική δραστηριότητα και ένα σύνολο αποτελεσμάτων, στο Merlin(γλώσσα τύπου Prolog), Oikos, ALF κανόνες τεχνητής νοημοσύνης (όπως planning και blakboard models)

Ενεργές Βάσεις δεδομένων ή οντοκεντρικές γλώσσες παράστασης όπου χρη-

σιμοποιούνται κανόνες της μορφής ECA(event condition action) για την περιγραφή της διαδικασίας και η εκτέλεση γίνεται μέσω Database triggers, όπως στο ADELE και στο SPADE

Δίκτυα Δραστηριοτήτων , Όπως για παράδειγμα τα Petri nets όπως στο SPADE

Διαδικαστικές γλώσσες, όπως στο ARCADIA [**?**]- IPSE2.5

Υβριδικοί τρόποι παράστασης όταν έχουμε συνδυασμό των διαφόρων προσεγγίσεων , όπως στο EPOS όπου μηχανισμοί κανόνων παραγωγής , συνδυάζονται με μηχανισμούς πυροδότητης, subtyping και δίκτυα δραστηριοτήτων.

2.3 Κατηγορίες μοντέλων

Στο βιβλίο των Finkelstein και Kramer [AFN94] αναφέρεται ότι ένα μοντέλο διαδικασίας αποτελείται από τρεις παραλλαγές:

Template Model: το μοντέλο εκφρασμένο τυπικά σε κάποια γλώσσα παράστασης διαδικασιών (PML), ή αλλιώς [**?**] Generic Model για τον ορισμό της διαδικασίας των projects ανάπτυξης λογισμικού.

Enactable Model, που είναι μοναδικά ορισμένη περίπτωση του πρώτου, με προσθήκη πληροφορίας εξειδικευμένης ως προς το συγκεκριμένο περιβάλλον (context) στο οποίο εφαρμόζεται και αρκετής για να γίνει εκτελέσιμο , ή αλλιώς Project-specific Model για τον ορισμό της διαδικασίας ενός συγκεκριμένου project ανάπτυξης λογισμικού

Enacting Model, που προκύπτει από ένα enactable model με σύνδεση με κάποιον διερμηνέα (interpreter) και συγκεκριμένη κατάσταση ενεργοποίησης

Επιπλέον, στο [**?**] αναφέρεται ότι υπάρχει ένα ανώτερο επίπεδο μοντέλου το meta model για τον ορισμό των στοιχείων μοντελοποίησης

Διάκριση των μοντέλων διαδικασιών γίνεται και ανάλογα με τους σκοπούς που εξυπηρετεί η περιγραφή της διαδικασίας. Έτσι διακρίνουμε τα προτρεπτικά (**prescriptive**) μοντέλα,που περιγράφουν τον τρόπο με τον οποίο θα έπρεπε να γίνει κάποια δουλειά, τα απαγορευτικά (**proscriptive**) μοντέλα, τα οποία χρησιμοποιούνται για τη περιγραφή του τί ΔΕΝ θα έπρεπε να γίνει, και τα περιγραφικά (**descriptive**) μοντέλα, που περιγράφουν αυτό που έχει γίνει. Τα αποτρεπτικά χρησιμοποιούνται ως επικουρικά των άλλων δύο κατηγοριών. Ο όρος descriptive χρησιμοποιείται και σαν αντίθετο του prescriptive για να περιγράψει τα μοντέλα εκείνα που λένε ΤΙ ακριβώς πρέπει να γίνει αλλά δεν ορίζουν σαφώς το πώς και πότε [DPS+93]. Συνήθως τη διαδικασία που πρόκειται να εφαρμοστεί εκφράζεται χρησιμοποιώντας προτρεπτικά μοντέλα, και προκειμένου να ελεγχθεί κατά πόσο ακολουθούνται πιστά και να δρομολογηθούν διαδικασίες βελτίωσής τους, απαιτείται και η χρήση των περιγραφικών. Είναι όμως ανοιχτό το πρόβλημα της

μεθόδων σύλληψης της περιγραφικής πληροφορίας και της ενσωμάτωσής των μετρήσεων σε αναπαραστάσεις διαδικασιών. Πώς δηλαδή μπορούμε από την περιγραφή της ιστορίας μιας διαδικασίας να ανατροφοδοτήσουμε την περιγραφή που την παρήγαγε με αυτόματο (ή τουλάχιστον μεθοδικό) τρόπο.

Στο ίδιο πλαίσιο γίνεται και ένας διαχωρισμός μεταξύ descriptive και active modeling, όπου το πρώτο θεωρείται ότι δίνει πληροφορία περισσότερο για μοντέλα που περιγράφουν τη συμπεριφορά διαδικασιών και οργανισμών ενώ το δεύτερο αναφέρεται στη χρήση των μοντέλων διαδικασιών για υπολογιστική υποστήριξη επιχειρήσεων και συστημάτων[Sno95].

Οι διαδικασίες λογισμικού συχνά αποτελούνται από διάφορες όψεις και επίπεδα λεπτομέρειας:

- ανάπτυξη λογισμικού, διαχείριση έργων, έλεγχος ποιότητας και διαχείριση διατάξεων.
- κύριες δραστηριότητες και υποδραστηριότητες, κύρια έγγραφα και υπο - έγγραφα
- κατανομή πόρων.

Στο [AK94] γίνεται διάκριση μεταξύ μοντέλου διαδικασίας και process definition document.Το πρώτο εκφράζει απλά την άποψη κάποιου μηχανικού για τη διαδικασία, ενώ το δεύτερο, είναι ένας οδηγός- καθοδηγητής για τη διαδικασία και περιέχει έννοιες χρήσιμες στον άνθρωπο που εκτελεί τη διαδικασία.

Κατηγορίες μοντέλων κατά Dowson

Σύμφωνα με την κατηγοριοποίηση του Dowson για τα μοντέλα διαδικασιών, υπάρχουν τρεις κατηγορίες τέτοιων μοντέλων:

1. Activity oriented (όπως το spiral [Boh88], τον αντικειμενοστρεφή κύκλο ζωής των συστημάτων (oo systems life cycle) [IH92]). Περιγράφονται με αλγοριθμικές γλώσσες, κατηγορικό λογισμό ή petri nets. Αυτά τα μοντέλα προέρχονται από αναλογία με τη λύση προβλημάτων, βασίζονται στην εύρεση και εκτέλεση ενός σχεδίου πράξεων που οδηγούν στη λύση.Είναι από τη φύση τους ακολουθιακά και παρέχουν ένα πλαίσιο για τη διαχείριση projects που εξελίσσονται με γραμμικό τρόπο. Χαρακτηριστικό τους μειονέκτημα είναι ότι η γραμμική άποψη είναι ακατάλληλη για διαδικασίες που έχουν αναχρησιμοποίηση, backtracking, και πρέπει να υποστηρίζουν παράλληλη εκτέλεση, όπως θεωρείται ότι είναι οι διαδικασίες παραγωγής λογισμικού.

2. Product oriented Χαρακτηριστικά παραδείγματα θεωρούνται η viewpoint oriented software developement του Finkelstein [NFK93, FKG90] όπου υποστηρί-

ζεται ότι η διαδικασία θα πρέπει να γνωρίζει την εσωτερική δομή και τη σημασιολογία των δεδομένων που χειρίζεται για να μπορεί να δώσει καθοδήγηση που να είναι ουσιαστική για το χρήστη [NFK93], και η γλώσσα μοντελοποίησης αντικειμένων και διαδικασιών (object and process modeling language - ESF report) Τα μοντέλα της κατηγορίας αυτής αναπαριστούν την διαδικασία ανάπτυξης μέσω της εξέλιξης του προϊόντος. Η άποψή τους για τη διαδικασία εξακολουθεί να έχει σαν κεντρική έννοια την δραστηριότητα ανάπτυξης, απλά δίνει έμφαση στη συσχέτισή μεταξύ της δραστηριότητας και της εξόδου της.

3. decision oriented Εδώ διαδοχικοί μετασχηματισμοί του προϊόντος θεωρούνται σαν συνέπειες αποφάσεων. Η τυπολογία των μοντέλων αυτής της κατηγορίας βασίζεται στο μοντέλο διαλόγων IBIS [CM88]. Η κατηγορία αυτή των μοντέλων είναι σημασιολογικά ισχυρότερη, γιατί εξηγούν όχι μόνο *πώς* αλλά και *γιατί* συμβαίνουν οι μετασχηματισμοί. Παρέχουν περισσότερο πλήρη γνώση για τη διαδικασία, που είναι χρήσιμη για την αναχρησιμοποίηση σχεδιαστικών αποφάσεων ανεπτυγμένων προιόντων αλλά και για backtracking purposes

Η έννοια του backtracking θεωρείται σημαντική γιατί σε περιπτώσεις όπου οι απαιτήσεις αλλάζουν είναι καλό να μπορούν να ανιχνευθούν οι αιτίες που οδήγησαν σε συγκεκριμένες σχεδιαστικές αποφάσεις.

2.4 Συστήματα

Παρουσιάζουμε παρακάτω σύντομα κάποια συστήματα που έχουν ενσωματωμένη δυνατότητα υποστήριξης της διαδικασία ανάπτυξης λογισμικού, τα οποία έχουν αναπτυχθεί τα τελευταία 4 χρόνια.

Θα αναφερθούμε στους φορμαλισμούς που υποστηρίζουν, το περιβάλλον και τον τρόπο με τον οποίο ενεργοποιούνται τα μοντέλα της διαδικασίας και θα εστιάσουμε στον τρόπο με τον οποίο το καθένα θεωρεί ότι παρέχει καθοδήγηση στο χρήστη.

ALF

Το μοντέλο της διαδικασίας στο ALF βασίζεται σε κανόνες παραγωγής και αφηρημένους τύπους δεδομένων. Σχηματίζεται από μια ιεραρχία μοντέλων υποστηριζόμενων διαδικασιών - MASPs (Model of Assisted Software Processes), επιτρέποντας έτσι περιγραφή σε πολλαπλά επίπεδα αφαίρεσης. Σε επίπεδο enaction model, ένα MASP είναι μία εξάδα *(Om,OPm,Rm,ORm,C)* όπου :

Om είναι το μοντέλο των αντικειμένων, τύπου ERA (Entity Relationship Attribute)

OPm είναι ένα σύνολο από τύπους τελεστών δηλαδή αφηρημένες περιγραφές (οικογένειες παρόμοιων) εργαλείων, που περιλαμβάνουν τις συνθήκες εφαρμογής τους, σύνδεση με συγκεκριμένα εργαλεία και συνθήκες που υποτίθεται ότι ισχύουν μετά την εφαρμογή τους.

Rm είναι ένα σύνολο κανόνων της μορφής Event Condition Action για να ορίζονται δραστηριότητες που θα πρέπει να ξεκινάνε αυτόματα σε κάποια κατάσταση της διαδικασίας.

ORm περιορισμοί σειράς με τη μορφή εκφράσεων μονοπατιών (path expressions) που καθορίζουν τη σειρά εκτέλεσης (σειριακά, παράλληλα ή εναλλακτικά), και *C= (characteristics)* περιορισμοί στις καταστάσεις της διαδικασίας με τη μορφή αναλλοίωτων (invariants).

Όταν οι παράμετροι αυτές των γενικών μοντέλων (των MASPs) παίρνουν τιμές, τότε τα μοντέλα γίνονται εκτελέσιμα (IMASPs) Έτσι, με την παραγωγή μίας περίπτωσης της MASP δημιουργείται ένα περιβάλλον εργασίας (αντικείμενα, σύνολο εργαλείων, διερμηνέας εντολών).

Η μετατροπή των μοντέλων σε συγκεκριμένα γίνεται είτε στατικά, πριν την εφαρμογή του μοντέλου, είτε δυναμικά, κατά τη διάρκεια της εκτέλεσης λαμβάνοντας υπόψιν πράγματα που έχουν συμβεί κατά την εκτέλεση (late binding). Στη συνέχεια δημιουργείται μία σύνοδος εργασίας work session που ονομάζεται ASP (Assisted Software Process) μεταξύ ενός χρήστη και ενός περιβάλλοντος εργασίας. Η επαφή χρήσης αποτελείται από εργαλεία που υποστηρίζουν τις πρωτοβουλίες του χρήστη (action tools), εργαλεία καθοδήγησης και εργαλεία πληροφόρησης του χρήστη για την εξέλιξη του συστήματος. Το όλο ΜΔ είναι κατανεμημένο μεταξύ εκτελέσιμων MASPs που ονομάζονται MINTs

COO

Πρόκειται για συνέχιση του ALF στο πανεπιστήμιο του Nancy. Αυτό το ερευνητικό πρόγραμμα έχει στόχο την ανάπτυξη ενός ενεργητικού πλαισίου υποστήριξης των διαδικασιών ανάπτυξης λογισμικού. Δίνει έμφαση ιδιαίτερα στο συντονισμό και την υποστήριξη της συνέπειας μεταξύ των διαφόρων εργασιών που εκτελούνται σε μια διαδικασία. Το χαρακτηριστικό του είναι ότι βασίζεται κυρίως σε ένα τυπικά ορισμένο μοντέλο δοσοληψιών.

Το μοντέλο της διαδικασίας εδώ είναι ένα σύνολο στόχων ψηλού επιπέδου και η περιγραφή του πώς αναλύονται σε υπό-στόχους. Επιπλέον υπάρχουν περιορισμοί ολοκλήρωσης που καθορίζουν πώς συντονίζονται οι υποστόχοι για την επίτευξη του ”πατέρα στόχου”. Κατάσταση εκτέλεσης της διαδικασίας θεωρείται η κατάσταση του δέντρου στόχων της και τα δεδομένα που έχουν παραχθεί από την στιγμή της ενεργοποίησής της.

Η γλώσσα του COO συντακτικά είναι υπογλώσσα της γλώσσας των MASPs αλλά χάρη σε νέες έννοιες έχει ισοδύναμη εκφραστική ικανότητα. Μια διαδικασία στο COO καθορίζεται από 1. τη λίστα των τυπικών της ορισμάτων, 2. τη δική της άποψη για τα δεδομένα, 3.τους ανθρώπους που σχετίζονται μ'αυτήν οι οποίοι επιλέγουν από τις δυνατές υποεργασίες ποια να εκτελέσουν και πότε, 4. τις προϋποθέσεις κάτω από τις οποίες μπορεί να ξεκινήσει (με τη μορφή εξισώσεων πρώτης τάξης που ορίζουν το χώρο εργασίας), 5. το στόχο της, που καθορίζει και τις συνθήκες τερματισμού της, 6.το σύνολο των δυνατών πρωταρχικών πράξεων που μπορούν να εκτελεστούν κότα τη διάρκειά της και 7. ένα σύνολο από περιορισμούς ολοκλήρωσης που συνεισφέρουν στο συγχρονισμό μεταξύ των υποδραστηριοτήτων

Όσον αφορά την εφαρμογή των μοντέλων διαδικασίας, το **COO** παρουσιάζει ιδιαιτερότητα βασίζοντας τους μηχανισμούς εκτέλεσης όχι σε ένα αμιγώς γνωσιοκεντρικό σύστημα, αλλά σε ένα ασφαλές μοντέλο δοσοληψιών το οποίο όταν η γνώση για τη διαδικασία είναι ανεπαρκής ορίζει ένα είδος default μοντέλου συγχρονισμού με χαλαρή serializability.

EPOS

Πρόκειται [1] για έναν αντικειμενοστρεφή πυρήνα υποστήριξης της διαδικασίας κατασκευής μεγάλων συστημάτων λογισμικού. Ταξινομείται στα υβριδικά συστήματα λόγω του ότι συνδυάζει πολλές διαφορετικές τυπολογίες. Παρέχει μια γλώσσα παράστασης διαδικασιών, τη SPELL, ένα αρχικό σχήμα διαδικασίας και ένα σύνολο εργαλείων υποστήριξης. [MJC94] [JC93].

Η SPELL (Software Process Evolutionary Language) είναι μια OO Language. Το μοντέλο της διαδικασίας αποτελείται από ένα σύνολο οντοκεντρικών κλάσεων και μετακλάσεων. Είναι ένα δίκτυο από περιγραφές δραστηριοτήτων.Οι τύποι των δραστηριοτήτων είναι κανόνες ενεργοποίησης, που περιγράφουν συνθήκες προ και κατόπιν της εκτελέσεως καθώς και την ανάλυση σε υποδραστηριότητες ή τον κώδικα που πρέπει να εκτελεστεί.

Στο **EPOS** ο χρήστης εφαρμόζει τα διάφορα εργαλεία αναγκαζόμενος ή υποβοηθούμενος από το σύστημα υποστήριξης της διαδικασίας. Το σύστημα υποστήριξης της διαδικασίας, αποτελείται από έναν διερμηνέα για τη SPELL, και έναν διαχειριστή εκτέλεσης, ο οποίος ερμηνεύει το δίκτυο δραστηριοτήτων. Η ερμηνεία του δικτύου δραστηριοτήτων συνίσταται στον έλεγχο πληροφορίας όπως (1) συνθήκες εκτέλεσης τύπων εργασίας, που ονομάζονται PRE-DYNAMIC

[1] EPOS = Expert System for Program and (''Og'' στα νορβηγικά System Development. Αναπτύχθηκε στο department of Computer Systems and Telematics του Norwegian Institute of Technology (IDT-NTH).

(2) αν η εργασία είναι ατομική, οπότε π.χ. απαιτείται η κλήση ενός compiler ή αν είναι υψηλού επιπέδου οπότε απαιτείται η κλήση του Planner (3) συνθήκες τέλους της εργασίας για να χειριστεί τα λάθη ή να ''καθαρίσει'' το χώρο.

Ο διαχειριστής εκτέλεσης συνδέεται με έναν broadcast message server για επικοινωνία με τα εξωτερικά εργαλεία. Ο *Planner* είναι μια συνάρτηση που καλείται από τον διαχειριστή εκτέλεσης η οποία εφαρμόζει τεχνικές τεχνητής νοημοσύνης (domain independent ai nonlinear planning by backword chaining and hierarchical decomposition) για να δημιουργήσει το δίκτυο υποεργασιών για τον αρχικό στόχο χρησιμοποιώντας και συγκεκριμένη γνώση του πεδίου εφαρμογής. Το ''σχήμα'' της διαδικασίας χρησιμεύει ως βάση γνώσης ενώ η δομή του προϊόντος λειτουργεί σαν περιγραφή της κατάστασης του κόσμου (World State Description).

Oikos

Οι στόχοι για τους οποίους φτιάχτηκε το **Oikos** [2] περιλαμβάνουν : τον ορισμό ενός συνόλου εννοιών και συμβολισμού για την παράσταση των διαδικασιών παραγωγής λογισμικού,

ένα τρόπο ιεράρχησης των δραστηριοτήτων ώστε ο κάθε πράκτορας που έχει κάποιο ρόλο στη διαδικασία να μπορεί να έχει μια εξειδικευμένη άποψη για τη διαδικασία όσο πλησιάζει στο ρόλο του,

και ένα μηχανισμό που να διευκολύνει τη μετατροπή του μοντέλου της διαδικασίας από αφηρημένο σε εκτελέσιμο παρέχοντας σύνδεση με άλλα εργαλεία και φροντίζοντας για την κατανομή του περιβάλλοντος ανάπτυξης.

Χρησιμοποιούνται δύο γλώσσες : μια για την περιγραφή του μοντέλου (Liimbo) και μία για την εφαρμογή του (Pate'. Και οι δύο είναι παράγωγα της ESP (Extended Shared Prolog)

Ταξινομείται στα υβριδικά συστήματα λόγω του ότι συνδυάζει διαφορετικές τυπολογίες. Στο **Oikos** το μοντέλο της διαδικασίας βασίζεται στο γνωστό μοντέλο του μαυροπίνακα και στη χρήση λογικού προγραμματισμού. Υπάρχει μια βάση γνώσης, ορατή σε όλους και καθένας από τους επιμέρους πράκτορες την εμπλουτίζει όταν κρίνει ότι πρέπει, αξιολογώντας ατομικά την κατάσταση, και αν μπορεί (αν πετύχει η ενημέρωση). Υπάρχει μια ιεραρχία από blackboards, που περιγράφουν αφηρημένα τη διαδικασία και εκλεπτύνεται (και γίνεται συγκεκριμένο) από πάνω προς τα κάτω, και ενημερώνεται με την προσθήκη λιγότερο αφηρημένων εννοιών,μη αιτιοκρατικά, concurrently, και κατανεμημένα. Το εκτελέσιμο μοντέλο προκύπτει από αυτές τις προσθήκες.

[2]Περιβάλλον για τη διευκόλυνση κατασκευής process centered software Development environments [AcC92] που αναπτύχθηκε στο πανεπιστήμιο της Πίζας.

ADELE

Το ADELE είναι δομημένο πάνω σε μια βάση δεδομένων που υποστηρίζει εκδόσεις, στην οποία αποθηκεύονται όλες οι συνιστώσες του περιβάλλοντος (προϊόντα και διαδικασίες). Στο **ADELE** ο process formalism (**TEMPO**) βασίζεται στις έννοιες του ”ρόλου”, από την άποψη ότι ένα αντικείμενο μπορεί να έχει διαφορετική συμπεριφορά σε διαφορετικές συνθήκες (περιβάλλοντα - καταστάσεις) και την έννοια της σύνδεσης που επιτρέπει συγχρονισμό και συντονισμό μεταξύ διαδικασιών. Η σύνδεση υποστηρίζεται με έναν διαχειριστή δραστηριοτήτων και ένα μηχανισμό προγραμματίσιμων πυροδοτήσεων που αυτόματα προωθεί την ανάγκη ενημερώσεων και ελέγχει την τήρηση των περιορισμών ακεραιότητας.

Στο **ADELE** η υποστήριξη της διαχείρισης της διαδικασίας γίνεται από έναν διαχειριστή δραστηριοτήτων και ένα διαχειριστή διαδικασίας (activity and process managment). Ο διαχειριστής διαδικασίας βασίζεται σε ένα μηχανισμό πυροδοτήσεων και κανόνες τύπου ECA (Event Condition Action) Ο διαχειριστής δραστηριοτήτων βασίζεται σε ένα μηχανισμό πυροδοτήσεων για να υποστηρίζει κάποιες επαναλαμβανόμενες δραστηριότητες που μπορούν να αυτοματοποιηθούν.

SPADE

Στο **SPADE** [3] η γλώσσα παράστασης των διαδικασιών ονομάζεται SLANG και δομείται πάνω σε μια επέκταση των petri nets. που ονομάζονται ER nets. Η τοπολογία των petri nets εκφράζει πληροφορία περί σχέσεων προτεραιότητας,συγκρούσεων και παραλληλισμού μεταξύ των δραστηριοτήτων. Η κατάσταση της διαδικασίας είναι σημειωμένη - κατανεμημένα στο δίκτυο, ενώ οι μεταβάσεις παριστάνουν δυνατά πραγματοποιήσεις Θεωρείται πλεονέκτημα της Slang η δυνατότητα ομοιόμορφης παράστασης διαφορετικών πλευρών της διαδικασίας, όπως η διαχείριση του ανθρώπινου δυναμικού, οι αλληλεπιδράσεις μεταξύ δραστηριοτήτων, και σχέσεις προτεραιότητας και χρονισμού. Ένα process model, στη SLANG είναι ένα ζεύγος από ένα σύνολο τύπων διαδικασιών και ένα σύνολο δεδομένων που ορίζονται με οντοκεντρική τεχνική σαν υποτύποι των token place, arc, transition που συναντάμε στα petri nets.

[3] Αναπτύχθηκε στο Plytechnico di Milano

Merlin

Το Merlin [4] κατατάσσεται στα συστήματα που χρησιμοποιούν τεχνικές βασισμένες σε κανόνες για την παράσταση και εφαρμογή (enactment) των διαδικασιών λογισμικού. Στο Merlin οι διαδικασίες ορίζονται με μια γλώσσα τύπου prolog σε τρία επίπεδα (project, process, kernel) και αποθηκεύονται σε μια βάση γνώσης. Τα στοιχεία που αποτελούν ένα ΜΔ στο Merlin είναι : activities, roles, documents, και resources. Η διαδικασία βασίζεται στον ορισμό των δραστηριοτήτων και των αντίστοιχων εγγράφων που αυτές χειρίζονται. Κάθε δραστηριότητα έχει ένα αριθμό από συνθήκες (προϋποθέσεις) που καθορίζουν τις υπευθυνότητες και τους ρόλους στη διαδικασία όπως ποιός είναι υπεύθυνος ή έχει πρόσβαση στο συγκεκριμένο έγγραφο καθώς και εξαρτήσεις με άλλες δραστηριότητες. Συγκεκριμένα, στο επίπεδο του project το μοντέλο της διαδικασίας αποτελείται από facts που ορίζουν την κατάσταση στο συγκεκριμένο πρόγραμμα (όπως π.χ. το όνομα του προγράμματος και των ατόμων που συμμετέχουν) Στο επίπεδο του process το ΜΔ αποτελείται από facts που ορίζουν τύπους εγγράφων, πιθανές καταστάσεις των εγγράφων και πιθανές μεταβάσεις καταστάσεων. Στο επίπεδο του Kernel το ΜΔ αποτελείται από κανόνες (rules) που χρησιμοποιούν τα παραπάνω facts και ουσιαστικά διαχειρίζονται τα περιβάλλοντα εργασίας.

Στο **Merlin** η επαφή χρήσης είναι δύο τύπων: ένα γραφικό περιβάλλον εργασίας γι'αυτούς που εφαρμόζουν τη διαδικασία και ένα για τη διαχειρίζονται των διαδικασιών. Η εκτέλεση κάθε εργασίας εισάγει καινούρια "facts" (σε ορολογία prolog) στο σύστημα πιθανότατα κάνοντας άλλες δραστηριότητες να είναι έτοιμες προς εκτέλεση. Η κύρια δουλειά της process engine είναι να εκτιμά - υπολογίζει όλες τις συνθήκες και να ενημερώνει όλα τα περιβάλλοντα εργασίας αντίστοιχα, κατόπιν διαταγής. Είναι ενδιαφέρον ότι έχει οριστεί μηχανισμός επίλυσης συγκρούσεων για το συγχρονισμό πολλαπλής πρόσβασης σε έγγραφα.

ARCADIA

Στόχος του ARCADIA [5] είναι η υποστήριξη παραγωγής περιβαλλόντων ανάπτυξης, ανάλυσης και διατήρησης μεγάλων και πολύπλοκων συστημάτων λογισμικού που έχουν μεγάλες απαιτήσεις αξιοπιστίας. Σ'αυτό το αμερικανικό

[4] Process Centered Software Development Environment [BSW92] που αναπτύχθηκε στο πανεπιστήμιο του Dortmund με την υποστήριξη των προγραμμάτων ESF (Eureka Software Factory) και PPOMOTER (ESPRIT III BRA project)

[5] ερευνητικό πρόγραμμα μελέτης εργαλείων και τεχνικών για τη βελτίωση της Software Engineering process Συμμετέχουν ερευνητές από τα :University of California, Irvine (UCI), University of Massachusetts at Amherst (UMass), University of Colorado at Boulder (CU), and Purdue University.

ερευνητικό πρόγραμμα, η διαδικασία περιγράφεται μέσω μιας γλώσσας προγραμματισμού, της APPL/A η οποία είναι επέκταση της Ada από την άποψη ότι προσφέρει προγραμματίσιμες συσχετίσεις μεταξύ αντικειμένων, δυνατότητα πυροδότησης στις λειτουργίες των σχέσεων και σύνθετα statements που της προσδίδουν transaction-like capabilities [see ProcessWall]

2.4.1 Εκτέλεση της διαδικασίας και καθοδήγηση

Εκτέλεση της διαδικασίας

Τα περισσότερα συστήματα που ενσωματώνουν μοντέλα διαδικασιών ισχυρίζονται [AFN94] ότι παρέχουν καθοδήγηση (guidance) και αποφεύγουν τον όρο επιβολή (enforcement) ενώ στην ουσία εστιάζουν κυρίως στην επιβολή περιορισμών με τη μορφή προτροπών και απαγορεύσεων. Κατά την άποψή μας, καθοδήγηση χρειάζεται όταν η ενεργοποίηση της διαδικασίας είναι μη αιτιοκρατική και κατά συνέπεια ο χρήστης πρέπει να αποφασίσει ή να επιλέξει κάποια πράγματα, εφόσον η ακολουθία των βημάτων δεν είναι προκαθορισμένη, και υπάρχουν εναλλακτικοί τρόποι εκπλήρωσης κάποιων στόχων.

Στα EPOS, ALF και ADELE όπου για την εκτέλεση της διαδικασίας απαιτείται ανθρώπινη επιρροή θεωρούμε ότι έχουμε μη αιτιοκρατικό τρόπο εκτέλεσης, ενώ αντίθετα στα Merlin και SPADE είναι αιτιοκρατικός.

Απαιτήσεις για μηχανισμούς εφαρμογής - εκτέλεσης

Συνοψίζοντας, για να μπορεί ένα σύστημα (ένα PCSEE) να έχει ενσωματωμένο σύστημα εφαρμογής μοντέλων διαδικασιών θα πρέπει να παρέχει:

▷ Μια βάση δεδομένων που να υποστηρίζει εκδόσεις και το μοντέλο των δεδομένων της να συνδυάζει οντοκεντρικές (OO) έννοιες αλλά και έννοιες από το μοντέλο οντοτήτων - συσχετίσεων - γνωρισμάτων (ERA)

▷ Άλλοι προσθέτουν ότι απαιτείται, για πραγματικές εφαρμογές, και ο συνδυασμός της βάσης αυτής με ένα σύστημα διαχείρισης διατάξεων (configuration management system)[MJC94]. Έτσι επιτυγχάνεται η δυνατότητα παρακολούθησης των διαφορετικών καταστάσεων του προϊόντος που μετασχηματίζεται από τη διαδικασία.

▷ Μηχανισμοί πυροδότησης παρέχουν υποστήριξη για επαναλαμβανόμενες και κατάλληλες για αυτοματοποίηση υποδραστηριότητες.

▷ Ειδικά για την περιγραφή και την ανάλυση των δραστηριοτήτων σε υποδραστηριότητες απαιτούνται εκτελέσιμες τυπολογίες (executable formalisms).

▷ Υποστήριξη για, πιθανώς φωλιασμένες, δοσοληψίες που διατηρούν για

μεγάλα χρονικά τον έλεγχο στα δεδομένα τους (Long transactions) για τον έλεγχο του χώρου εργασίας (workspace control) και για όψεις βάσεων δεδομένων ή "υποβάσεων" (subdatabases) όπως γίνεται στο Merlin (transactiopn concept) και στο EPOS (long and nested transactions) και μηχανισμοί επίλυσης ή αποφυγής των συγκρούσεων που μπορεί να προκύψουν από ταυτόχρονη πρόσβαση του ίδιου εγγράφου από χρήστες ή από τον μηχανισμό εφαρμογής της διαδικασίας.

▷ Σύστημα διαχείρισης πολλών περιβαλλόντων εργασίας (workspace management) απαιτείται για την υποστήριξη επικοινωνίας και συγχρονισμού μεταξύ ομάδων που μοιράζονται αντικείμενα (όπως στο ALF, ADELE και EPOS)

Σαν γενική παρατήρηση μπορούμε να αναφέρουμε ότι η επειδή η διαδικασία ανάπτυξης λογισμικού είναι μια διαδικασία με πολλούς χρήστες, παρουσιάζεται η τάση για κατανεμημένους μηχανισμούς εφαρμογής των μοντέλων.

2.5 IPSW 6

Αξίζει να αναφερθεί το 6ο και 7ο International Workshop on Sofftware Process όπου ορίστηκε ένα χαρακτηριστικό παράδειγμα προβλήματος μοντελοποίησης διαδικασίας λογισμικού, το οποίο αποτελεί ένα πλαίσιο σύγκρισης των διαφόρων τυπολογιών που αφορούν μοντέλα διαδικασιών [KFF+90].

Το βασικό πρόβλημα αναφέρεται είτε σε ένα αρκετά προχωρημένο στάδιο της φάσης σχεδιασμού, ή στη φάση της συντήρησης του λογισμικού. Το πρόβλημα εστιάζει στη σχεδίαση, υλοποίηση, έλεγχο (testing) και διαχείριση μιας τοπικής αλλαγής στο σύστημα λογισμικού. Ξεκινάει με τον προγραμματισμό της αλλαγής από τον διαχειριστή του προγράμματος και τελειώνει όταν η αλλαγή έχει δοκιμαστεί και έχει περάσει τα τεστ επιτυχώς.

Αξίζει να σημειωθεί ότι όλα τα παραπάνω συστήματα έχουν πετύχει να εκφράσουν το τυπικό προβλημα του ISPW-6 στο φορμαλισμό τους.

Κεφάλαιο 3

Οντοκεντρικές Μέθοδοι Σχεδίασης και Ανάλυσης

Κεντρική έννοια στην εργασία μας είναι η αναλογική καθοδήγηση. Αναλογική καθοδήγηση ονομάζουμε την υποστήριξη της λήψης απόφασης σε καταστάσεις διλημμάτων μέσω παρουσίασης ανάλογων παραδειγμάτων. Θέλουμε να διερευνήσουμε τη χρησιμότητα μιας τέτοιου τύπου κααθοδήγησης στην κατασκευή μεγάλων συστημάτων λογισμικού. Στην ανάλυση και προδιαγραφή τέτοιων συστημάτων , συνηθίζεται να εφαρμόζονται οντκεντρικές μεθοδολογίες, για λόγους πολυπλοκότητας και εεπεκτασιμότητας. Κοινό χαρακτηριστικό των μεθόδων αυτών είναι η παρουσία γενικών κατευθύνσεων με πολαπλές επιλογές. Η παρουσίαση ανάλογων παραδειγμάτων από εφαρμογές αυτών των γενικών κατευθύνσεων μπορεί να συμβάλει στην καλύτερη εφαρμογή τους.

Σ'αυτό το κεφάλαιο θα δούμε σύντομα κάποιες χαακτηριστικές οντοκεντρικές μεθοδολογίες , δηλαδή μεθόδους με οντοκεντρική φιλοσοφία.

3.1 Συνοπτική θεώρηση των μεθόδων OOA- OOD

Με τον όρο Object-Oriented Methodology αναφερόμαστε σε μεθοδολογίες ανάλυσης και σχεδίασης που θεωρούν το υπό κατασκευή σύστημα σαν ένα σύνολο αντικειμένων (objects) Η θεώρηση αυτή συμβάλλει ουσιαστικά στην αντιμετώπιση της πολυπλοκότητας που είναι εγγενές χαρακτηριστικό των μεγάλων συστημάτων. Υπάρχουν πολλές συγκρίσεις τέτοιων μεθοδολογιών, στη βιβλιογραφία , αλλά δεν υπάρχει γενικά αποδεκτός τρόπος αξιολόγησής τους. Μπορούμε να τις διακρίνουμε σε τρεις βασικές κατηγορίες ανάλογα με τη βασική τεχνική που χρησιμοποιούν για να διαμερίζουν το σύστημα στις συνιστώσες - αντικείμενα.

Οι κατηγορίες αυτές είναι : α. Data-Centered, όπως η OMT (Object Modeling Techic) μέθοδος μοντελοποίησης αντικειμένων του Rumbaugh β. Scenario-Based, όπως το Jacobson Objectory γ. δομικές Structural, όπως η μέθοδος του Booch, από την άποψη ότι δεν υπάρχει αυστηρά προκαθορισμένη διαδικασία, αλλά είναι πολύ καλά ορισμένη η δομή των προϊόντων της διαδικασίας. Και οι τρεις προσεγγίσεις συμβάλλουν στην κατανόηση των απαιτήσεων και στη σχεδίαση του συστήματος.

Ενδεικτικά περιγράφουμε παρακάτω τρεις αντιπροσωπευτικές μεθοδολογίες.

Rumbaugh - Object Modeling Technique

Η κατασκευή ενός συστήματος με OMT περνάει από τρεις διαδοχικές φάσεις. Την ανάλυση, τη σχεδίαση και την υλοποίηση. Σύμφωνα με αυτή τη μέθοδο, το προς κατασκευή σύστημα θεωρείται ότι έχει τρεις όψεις

- το **μοντέλο αντικειμένων** (Object Model) για την περιγραφή της στατικής δομής του συστήματος, που περιλαμβάνει τις κλάσεις, τις σχέσεις μεταξύ τους τα γνωρίσματα και τις λειτουργίες που τις χαρακτηρίζουν. Ο συμβολισμός που χρησιμοποιείται για να περιγραφεί ένα τέτοιο μοντέλο στη φάση της ανάλυσης είναι μια επέκταση του οντοτήτων- συσχετίσεων (Entity-Relationship) για την παράσταση των παραπάνω στοιχείων.

- το **δυναμικό μοντέλο** (Dymanic Model) για την παράσταση της δυναμικής συμπεριφοράς κάθε κλάσης. Εδώ χρησιμοποιούνται διαγράμματα αυτομάτων (μηχανές καταστάσεων). Οι μεταβάσεις οφείλονται σε συμβάντα (events) και οι καταστάσεις είναι αφαιρέσεις των γνωρισμάτων και των σχέσεων των αντικειμένων Τα διαγράμματα αυτά δείχνουν πως τα αντικείμενα αλλάζουν την κατάστασή τους ή εκτελούν λειτουργίες.

- το **λειτουργικό μοντέλο** (Functional Model) δείχνει τη συμπεριφορά του συστήματος χρησιμοποιώντας διαγράμματα ροής δεδομένων (DataFlow Diagrams, για συντομία DFD). Στο DFD οι πηγές, καταβόθρες, αποθήκες δεδομένων και οι είσοδοι έξοδοι στους κόμβους παριστάνουν αντικείμενα ενώ οι κόμβοι παριστάνουν διεργασίες.

Τα παραπάνω συστήματα κτίζονται με τη σειρά στη φάση της ανάλυσης. Φυσικά επιτρέπονται οι επαναλήψεις, για την ολοκλήρωση των μοντέλων.

Στη φάση της σχεδίασης, διακρίνουμε δυο υπο-φάσεις: τη σχεδίαση του συστήματος και τη σχεδίαση των αντικειμένων. Για τη σχεδίαση χρησιμοποιούμε τα μοντέλα που προκύπτουν από την ανάλυση. Στη σχεδίαση του συστήματος ασχολείται με τη δομή (την αρχιτεκτονική) του συστήματος, εντοπίζοντας τα υποσυστήματα

Jacobson's Objectory Method

Η μέθοδος αυτή βασίζεται στην έννοια της περίπτωσης χρήσης (use case) που είναι ένας διάλογος μεταξύ του συστήματος και του χρήστη, που εκτελείται για την ικανοποίηση κάποιου στόχου. Στη μέθοδο αυτή διακρίνονται τρεις φάσεις:

- requirements Ξεκινώντας από την περιγραφή των απαιτήσεων σε φυσική γλώσσα, σ'αυτή τη φάση κατασκευάζεται ένα μοντέλο περιπτώσεων χρήσης, ένα μοντέλο αντικειμένων του πεδίου εφαρμογής και μια περιγραφή της επαφής χρήσης του συστήματος.

- analysis σ'αυτή τη φάση κατασκευάζεται ένα μοντέλο πάλι και η περιγραφή των υποσυστημάτων. Θεωρητικά εδώ παράγεται μια λεπτομερής ιδανική περιγραφή του συστήματος Το μοντέλο σ'αυτή τη φάση, είναι ένα είδος ER model. Παριστάνονται αντικείμενα, κλάσεις και σχέσεις. Οι σχέσεις είναι τύπου κληρονόμησης και επέκτασης. Επίσης μεταξύ αντικειμένων μπορούν να ορισθούν σχέσεις τύπου acquaintance, που σημαίνει ότι το ένα αναφέρεται στο άλλο Ειδικές περιπτώσεις της θεωρούνται οι σχέσεις συγκρότησης και επικοινωνίας. Τα αντικείμενα μπορεί να είναι αντικείμενα ελέγχου ή οντότητες, ή αντικείμενα επαφής (interface).

- construction Σ'αυτή τη φάση κατασκευάζεται ένα μοντέλο τμημάτων, διαγράμματα (χρονικής) αλληλεπίδρασης και μοντέλο επαφής ζεύξης των τμημάτων Προαιρετικά κατασκευάζεται και ένα μοντέλο καταστάσεων (state machine) για τα αντικείμενα.

Booch Object Oriented Analysis and Design

Η μέθοδος σχεδίασης του Booch [Boo93] ορίζει δύο αλληλοσυμπληρωνόμενες όψεις για τη μέθοδο: την μικροδιαδικασία και την μακροδιαδικασία. Η μακρο διαδικασία είναι τυπική και ακολουθιακή, ακολουθώντας τη γενικότερη κατηγορία των μοντέλων καταρράκτη (waterfall models) Περιλαμβάνει φάσεις :
1. Καθιέρωση των βασικών προδιαγραφών (Conceptualization)
2. Ανάπτυξη μοντέλου της επιθυμητής συμπεριφοράς (ανάλυσης απαιτήσεων)
3. Δημιουργία αρχιτεκτονικής (Σχεδίαση)
4. Εξέλιξη της υλοποίησης (Evolution)
5. Διαχείριση της εξέλιξης κατόπιν παραδόσεως του προϊόντος(maintainance)

Όσον αφορά τη μικροδιαδικασία, μπορούμε να πούμε ότι συνίσταται στα εξής 4 βασικά βήματα, τα οποία είναι μη τυποποιημένα, και επαναλαμβάνονται σπειροειδώς.

1. Αναγνώριση κλάσεων και αντικειμένων. Σ'αυτή τη φάση οι σχεδιαστές αναλύουν τις προδιαγραφές του συστήματος. Στόχος είναι να εντοπιστούν

οι βασικές κλάσεις και αντικείμενα που μπορούν να προέρχονται από οντότητες του πεδίου εφαρμογής του συστήματος (problem domain) ή να είναι μηχανισμοί που απαιτούνται για την ικανοποίηση των προδιαγραφών Το αποτέλεσμα αυτής της φάσης είναι ένα σύνολο από υποψήφιες κλάσεις και αντικείμενα.

2. Καθορισμός του σημασιολογικού περιεχομένου (της σημασίας - semantics) των κλάσεων και αντικειμένων. Εδώ εξακριβώνεται ποιές από τις υποψήφιες κλάσεις μπορούν να οριστούν τελικά στη σχεδίαση των προδιαγραφών, και για αυτές που επιλέχτηκαν καθορίζονται τα πεδία τους και οι λειτουργίες τους

3. Καθορισμός συσχετίσεων μεταξύ κλάσεων. εδώ, καθορίζονται σχέσεις χρήσης, κληρονόμησης, γενίκευσης μεταξύ κλάσεων.

4. Υλοποίηση κλάσεων. Εδώ καταρχήν επιλέγονται κάποιοι μηχανισμοί και στη συνέχεια υλοποιούνται συγκεκριμένοι μηχανισμοί και προγραμματιστικές δομές για την υλοποίηση των κλάσεων και των αντικειμένων. Σ'αυτή τη φάση παράγονται λειτουργικά τμήματα του συστήματος τα οποία μπορούν να ομαδοποιηθούν σε υποσυστήματα.

Η μεθοδολογία του Booch παρέχει γενικές οδηγίες και υποδείξεις για το πως εφαρμόζονται τα παραπάνω βήματα, και ένα πολύ καλά ορισμένο μοντέλο των ενδιάμεσων προϊόντων. Επειδή η μέθοδος καθοδήγησης που προτείνουμε στοχεύει στην υποστήριξη των σχεδιαστικών αποφάσεων με βάση αναλογίες που εντοπίζονται στο προϊόν σε συνδυασμό με την συσχέτιση της τεκμηρίωσης σχεδιαστικών αποφάσεων μ'αυτό, επιλέγουμε τη μέθοδο αυτή για το παράδειγμα καθοδήγησής μας.

Αναλυτικότερα θα την εξετάσουμε στο κεφάλαιο 6, υπό το πρίσμα ενός αποφασεοκεντρικού μοντέλου διαδικασιών.

Κεφάλαιο 4

Ανάλυση ομοιότητας

Σ'αυτό το κεφάλαιο, θα δούμε σύντομα το μοντέλο αναλογικής ομοιότητας που χρησιμοποιούμε, και'αρχήν όπως ορίστηκε και στη συνέχεια θα περιγράψουμε τον τρόπο με τον οποίο το χρησιμοποιούμε για να πετύχουμε το στόχο μας.

Πρώτα θα παρουσιάσουμε σύντομα τη γλώσσα παράστασης γνώσης που χρησιμοποιούμε,έπειτα θα μιλήσουμε για το μοντέλο αναλογικής ομοιότητας διαισθητικά, και τέλος θα δώσουμε ένα παράδειγμα εντοπισμού αναλογιών σε περιγραφές.

4.1 Γλώσσα παράστασης γνώσης Telos

Η παράσταση των περιγραφών που χειριζόμαστε γίνεται στη γλώσσα Telos [MBJK90]. Η Telos είναι μια οντοκεντρική γλώσσα παράστασης γνώσης. Υποστηρίζει τρεις μηχανισμούς αφαίρεσης (ταξινόμηση, γενίκευση και απόδοση γνωρισμάτων). Η Telos χειρίζεται τις οντότητες και τα γνωρίσματα ομοιόμορφα, σαν αντικείμενα που έχουν ίσα δικαιώματα.

Τα αντικείμενα μπορούν να ταξινομούνται σε μία ή περισσότερες κλάσεις οι οποίες εισάγουν διαφορετικά είδη γνωρισμάτων με τα οποία φτιάχνονται οι περιγραφές. Οι κλάσεις , θεωρούνται κι αυτές οντότητες και μπορούν να ταξινομούνται σε κλάσεις. Μία οντότητα που ταξινομείται σε κάποια κλάση ονομάζεται περίπτωση (instance) της κλάσης αυτής.

Ένα γνώρισμα αποτελείται από μια πηγή (source) , μια ετικέτα (label) και έναν προορισμό (destination). Τα γνωρίσματα,αφού αντιμετωπίζονται σαν οντότητες, ταξινομούνται επίσης σε μία ή περισσότερες κλάσεις γνωρισμάτων που ονομάζονται κατηγορίες.

Οι κλάσεις μπορούν να σχετίζονται μεταξύ τους με σχέσεις γενίκευσης (IsA) , οι οποίες επιβάλλουν αυστηρή κληρονόμηση γνωρισμάτων. Με σχέση

γενίκευσης σχετίζονται κλάσεις του ίδιου επιπέδου ταξινόμησης και η σχέση αυτή έχει σημαντική υποσυνόλου.

Στην Telos μπορούμε να παριστάνουμε συγκροτήσεις χρησιμοποιώντας το μηχανισμό απόδοσης γνωρισμάτων, χωρίς την ανάγκη ύπαρξης ειδικού μηχανισμού συγκρότησης.

Το πλαίσιο παράστασης της Telos μπορεί να χρησιμοποιηθεί για διαφορετικά μοντέλα περιγραφής απαιτήσεων (DFDs, ERDs,STDs) ως εξής : τα μοντέλα αυτά κατ'αρχήν περιγράφονται σαν μεταμοντέλα στη γλώσσα , και οι περιγραφές των απαιτήσεων εκφράζονται σαν περιπτώσεις αυτών των μετα μοντέλων.

Η Telos θεωρείται κατάλληλη για την παράσταση περιγραφών επειδή υποστηρίζει πολύπλοκες, φωλιασμένες, και αναδρομικές δομές, πολλαπλή και ιεραρχική δόμηση αντικειμένων, σχέσεις καθορισμένου τύπου με γνωρίσματα και ομαδοποιήσεις σε πολλαπλούς και πιθανώς επικαλυπτόμενους χώρους παρέχοντας διαφορετικές όψεις των οντοτήτων [ea92].

4.2 Αναλογική ομοιότητα περιγραφών

Η ομοιότητα μεταξύ δύο περιγραφών που έχουν παρασταθεί σε Telos ορίζεται σαν συνάρτηση που παίρνει τιμές στο διάστημα [0,1]. Καθορίζεται από το αντίστροφο των εννοιολογικών αποστάσεων μεταξύ των περιγραφών.

Η εννοιολογική απόσταση δύο περιγραφών υπολογίζεται συνοψίζοντας τις επί μέρους αποστάσεις τους (ταύτισης, γενίκευσης, ταξινόμησης, γνωρισμάτων).

Η *απόσταση ταύτισης* καθορίζει αν δύο ονόματα - δοσμένα από το χρήστη- αναφέρονται στο ίδιο αντικείμενο ή όχι.

Η *απόσταση ταξινόμησης* εκφράζει τις διαφορές μεταξύ των δύο αντικειμένων που προκύπτουν από την ταξινόμησή τους σε διαφορετικές κλάσεις. Οι μη κοινές κλάσεις των αντικειμένων συμβάλλουν διαφορετικά στην απόσταση ταξινόμησης ανάλογα με τη σημασία της κατηγοριοποίησης που εκφράζουν. Η σχετική σπουδαιότητα μιας κλάσης είναι μια φθίνουσα συνάρτηση του βάθους της κλάσης στους γράφους ταξινόμησης και γενίκευσης που συμμετέχει.

Η *απόσταση γενίκευσης* εκφράζει διαφορές όπως έχουν συλληφθεί κατά τον προσδιορισμό των μη κοινών υπερκλάσεων. Οι μη κοινές υπερκλάσεις συμβάλλουν στην απόσταση γενίκευσης με τον ίδιο τρόπο που οι μη κοινές κλάσεις συμβάλλουν στην απόσταση ταξινόμησης.

Η *απόσταση γνωρισμάτων* εκφράζει τις διαφορές που εμφανίζονται στο επίπεδο της απόδοσης γνωρισμάτων. Ο υπολογισμός αυτής της απόστασης περιλαμβάνει την αποκατάσταση μιας μερικής αντιστοιχίας των γνωρισμάτων των δύο αντικειμένων. Σ'αυτή την αντιστοιχία, κάθε ένα από τα γνωρίσματα του

ενός αντικειμένου μπορεί να αντιστοιχεί σε ένα ή κανένα γνώρισμα του άλλου αντικειμένου. Έτσι τα γνωρίσματα καθενός από τα αντικείμενα που συγκρίνουμε διακρίνονται σε μοναδικά και ανάλογα. Προκειμένου να μπορούν να αντιστοιχηθούν δύο γνωρίσματα μεταξύ τους θα πρέπει να ανήκουν σε κάποια κοινή κλάση γνωρισμάτων (**αρχή της σημασιολογικής ομοιογένειας**). Στην περίπτωση που είναι δυνατές περισσότερες από μία αντιστοιχίσεις, χρησιμοποιείται ένα κριτήριο ελάχιστης απόστασης για να επιλεγεί η βέλτιστη από αυτές για να επιλεγεί η αντιστοίχιση εκείνη που δίνει την ελάχιστη συνολική απόσταση.

Αυτή η απόσταση είναι η συγκρότηση των αποστάσεων των ζευγών γνωρισμάτων που προκύπτουν κατά την αντιστοίχιση. Ο υπολογισμός των αποστάσεων μεταξύ δύο αντιστοιχίσιμων γνωρισμάτων γίνεται θεωρώντας τα ως οντότητες. Έτσι ο υπολογισμός της απόστασης γνωρισμάτων είναι αναδρομικός: δημιουργεί βέλτιστες αντιστοιχίσεις στα γνωρίσματα των γνωρισμάτων κ.ο.κ. Τα γνωρίσματα των αντικειμένων παίζουν λιγότερο ή περισσότερο σημαντικό ρόλο στην απόσταση γνωρισμάτων τους σύμφωνα με ένα συντελεστή σπουδαιότητας (salience). Ο συντελεστής αυτός καθορίζεται από τους εξής παράγοντες: την ικανότητα του γνωρίσματος να διακρίνει το αντικείμενο μεταξύ άλλων όσο αφορά σε ιεραρχίες γενίκευσης (*characterici*ty= χαρακτηριστικότητα), το πόσο σημαντικό είναι το γνώρισμα για τη δομή και τη συμπεριφορά του αντικειμένου (*abstractness* = βαθμός αφαίρεσης), και την ικανότητα του γνωρίσματος να καθορίζει την τιμή άλλων αντικειμένων (*determinance*= προσδιοριστικότητα). Περιορίζοντας το βάθος της αναδρομής παίρνουμε προσεγγιστικές τιμές για την απόσταση γνωρισμάτων.

Ο παραπάνω ορισμός εννοιολογικής απόστασης εξαρτάται μόνο από τον τρόπο αναπαράστασης του μοντέλου των δεδομένων, και τη συγκεκριμένη περιγραφή που έχει αποθηκευτεί χωρίς να προϋποθέτει υποκειμενικές εκτιμήσεις των εννοιολογικών αποστάσεων. Αυτό είναι σημαντικό πλεονέκτημα, δεδομένου ότι τέτοιες εκτιμήσεις είναι δύσκολο να παρέχονται συνεχώς και επιπλέον διαφέρουν από άτομο σε άτομο. Επιπλέον, δεν προϋποθέτει ομοιομορφία των μοντέλων περιγραφής των συγκρινόμενων περιγραφών και ο υπολογισμός γίνεται αποδοτικότερα όσο πιο πολύπλοκες γίνονται οι περιγραφές[SC95].

4.3 Αναλογική ομοιότητα στα πλαίσια μοντέλων διαδικασιών

Ένα από τα βασικά συστατικά ενός μοντέλου διαδικασίας είναι το προϊόν που παράγεται. Στο μεταμοντέλο διαδικασιών που θα χρησιμοποιήσουμε, το προϊόν χρησιμεύει για την περιγραφή της περίστασης, η οποία είναι συστατικό στοιχείο του περιβάλλοντος απόφασης.

Προκειμένου να παρέχουμε καθοδήγηση με τη μορφή κατάλληλων παραδειγμάτων, πρέπει να μπορούμε να εντοπίζουμε τα παραδείγματα αυτά με βάση την ομοιότητά τους προς την τρέχουσα περίσταση. Συγκεκριμένα για τους σκοπούς της εργασίας μας χρειαζόμαστε να μπορούμε να εντοπίζουμε αναλογίες μεταξύ περιβαλλόντων απόφασης και ειδικότερα μεταξύ των συνιστωσών περίστασης τους. Επιπλέον μας ενδιαφέρει να μπορούμε να δούμε ποια από τα στοιχεία που απαρτίζουν τις συγκρινόμενες καταστάσεις είναι ανάλογα μεταξύ τους. Αυτή η πληροφορία χρησιμεύει για να εκτιμήσουμε κατά πόσο οι ομοιότητα των δύο καταστάσεων βασίζεται σε συστατικά τους που εμείς τα θεωρούμε σημαντικά. Επιπλέον μπορεί να χρησιμεύσει για να δούμε κατά πόσο μπορούμε να εκφράσουμε ανάλογα επιχειρήματα που να στηρίζονται σε ανάλογα τμήματα προϊόντος.

Πιστεύουμε ότι μια καλή επιλογή για το σκοπό μας, είναι να παριστάνουμε τις περιστάσεις σαν σύνθετα αντικείμενα που τα συστατικά τους είναι οντότητες που ανήκουν στην ιεραρχία ταξινόμησης που έχει κορυφή την κλάση *Προϊόν (Product).* Όπως αναφέρθηκε στην παράγραφο για την Telos(4.1), οι συγκροτήσεις οντοτήτων και σχέσεων μεταξύ οντοτήτων μπορούν, με το μηχανισμό απόδοσης γνωρισμάτων, να περιγραφούν σαν οντότητες που τα γνωρίσματά τους παριστάνουν τα αντικείμενα-συνιστώσες.

Έτσι υπολογίζοντας την ομοιότητα μεταξύ περιστάσεων, αποκαθίσταται μια αντιστοίχιση μεταξύ των συστατικών τους. Δηλαδή, ο υπολογισμός της ομοιότητας με το μοντέλο που περιγράφηκε προηγουμένως δίνει για τις συγκρινόμενες καταστάσεις τις αντιστοιχίσεις των προϊόντων που τις αποτελούν. Αυτό σημαίνει ότι μπορούμε να δούμε όχι μόνο ποιό ίχνος από την εκτέλεση της διαδικασίας είναι ανάλογο με την περίπτωση του διλήμματος που αντιμετωπίζουμε κάθε φορά, αλλά επιπλέον να δούμε ως προς ποιές συνιστώσες της περίστασης είναι ανάλογο.

Οι αντιστοιχίσεις αυτές , εξαιτίας της αναδρομής στον υπολογισμό της απόστασης γνωρισμάτων, έχουν λάβει υπόψιν τους τη συνολική δομή του προϊόντος και γι'αυτό είναι αξιόπιστες.

4.4 Παράδειγμα εντοπισμού αναλογιών

Εδώ θα δούμε αναλυτικά ένα παράδειγμα που δείχνει τον εντοπισμό αναλογιών μέσω ανάλυσης ομοιότητας μεταξύ *περιστάσεων*.

Οι περιστάσεις που συγκρίνουμε περιγράφουν περιπτώσεις μερικώς περιγεγραμμένων διαγραμμάτων ροής δεδομένων. Οι δύο περιστάσεις παριστάνονται σαν περιπτώσεις (instances) ενός μεταμοντέλου διαδικασιών σε Telos.

Μια περίσταση είναι μια συγκρότηση από τμήματα προϊόντος, δηλαδή ένα σύνθετο αντικείμενο με γνωρίσματα που έχουν προορισμό περιπτώσεις του προϊόντος.

Σύμφωνα με το μεταμοντέλο διαδικασιών, υπάρχει κατ'αρχήν ένα μοντέλο προϊόντος περίπτωση του οποίου αποτελεί το μοντέλο για διαγράμματα ροής δεδομένων (DFD) που μας απασχολεί εδώ. Ένα DFD, λοιπόν, περιγράφεται από *επεξεργασίες* (processes) τις *εισόδους* και *εξόδους* τους, τις *μεταβάσεις καταστάσεων* που προκαλούν σε κάποιον *πόρο* και τους *πράκτορες* που σχετίζονται μ'αυτές.

Η περίσταση που αφορά τα μερικώς περιγεγραμμένα διαγράμματα ροής δεδομένων (PartiallyDescribedDFD) είναι περίπτωση της κλάσης *Περίσταση (Situation)* του μεταμοντέλου, και το γνώρισμά της dfdEntities είναι περίπτωση του γνωρίσματος dependsOn που έχει αφετηρία την κλάση Situation και προορισμό την κλάση *Product*.

Στο Telos μοντέλο μας οι περιστάσεις που θα συγκρίνουμε είναι οντότητες, περιπτώσεις της κλάσης PartiallyDescribedDFD που είναι περίπτωση της κλάσης Situation (περίσταση).

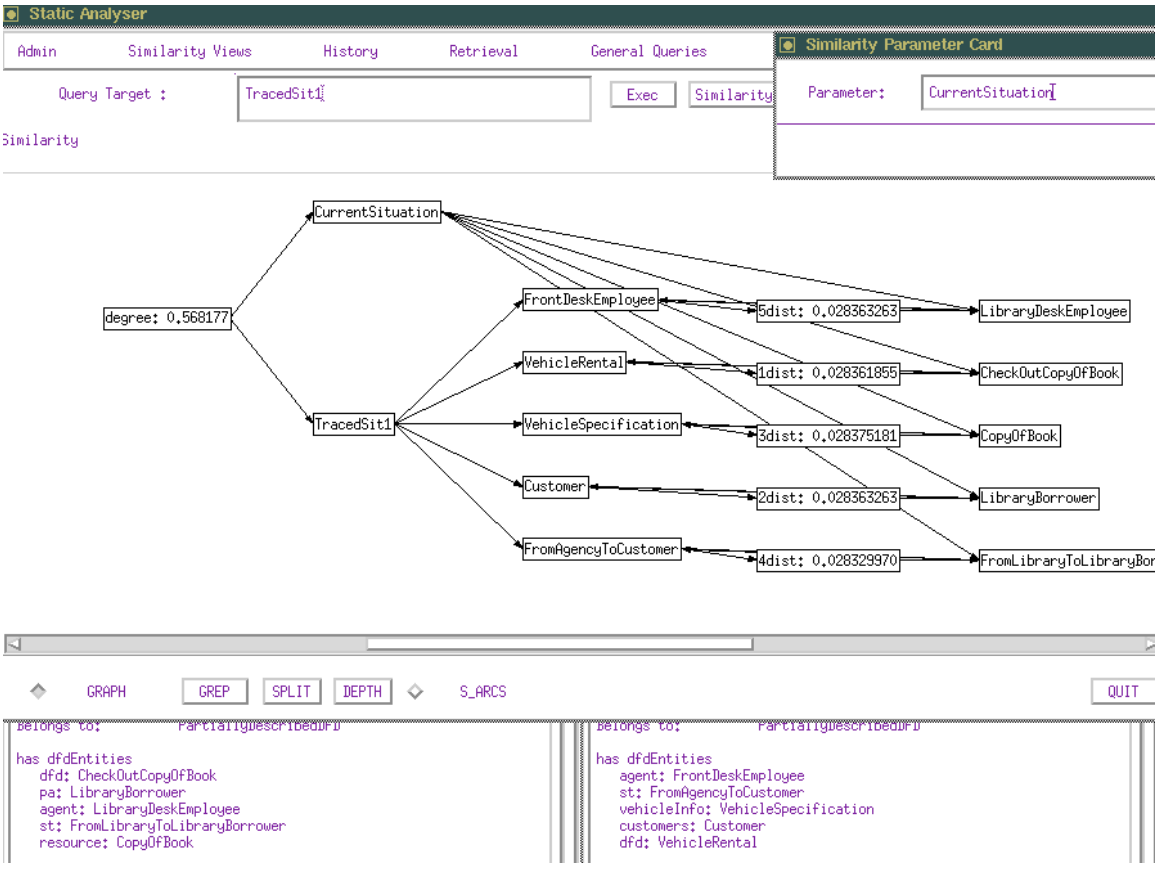
Η πρώτη περίσταση (CurrentSituation) αφορά την κατάσταση στην οποία βρίσκεται η διαδικασία περιγραφής του διαγράμματος ροής δεδομένων για την δοσοληψία δανεισμού βιβλίου όταν η πρόθεση είναι να οριστεί η είσοδος στη συγκεκριμένη δοσοληψία. Σ'αυτή την περίσταση, έχουν οριστεί η δοσοληψία *CheckOutCopyfBook*, οι πράκτορες *LibraryDeskEmployee και LibraryBorrower*, ο πόρος *CopyOfBook* και η μετάβαση κατάστασης που προκαλεί η επεξεργασία του DFD, *FromLibraryToLibraryBorrower*. Όλες αυτές οι οντότητες συγκροτούν την περίσταση CurrentSituation.

Η δεύτερη περίσταση (TracedSit1) αφορά την κατάσταση στην οποία βρέθηκε η διαδικασία περιγραφής του διαγράμματος ροής δεδομένων για την δοσοληψία δανεισμού οχήματος, και είναι περίπτωση του ίδιου περιβάλλοντος απόφασης (δηλαδή έχει την ίδια πρόθεση). Εδώ έχουν οριστεί η δοσοληψία *VehicleRental*, οι πράκτορες *FrontDeskEmployee και Customer* , ο πόρος *VehicleSpecification* και η μετάβαση κατάστασης που προκαλεί η επεξεργασία

του DFD, *FromAgencyToCustomer*.

Όλες αυτές οι οντότητες συγκροτούν την περίσταση TracedSit1. Η περιγραφή των περιστάσεων με βάση τις συνιστώσες τους σε Telos φαίνεται στο κάτω μέρος του σχήματος 4.1.

Ο υπολογισμός της ομοιότητας των CurrentSituation και TracedSit1 , ανάγεται στον υπολογισμό μιας αντιστοίχισης των γνωρισμάτων τους που είναι βέλτιστη ως προς το κριτήριο ελάχιστης συνολικής απόστασης. Σύμφωνα με την αρχή της σημασιολογικής ομοιογένειας, αυτή η αντιστοίχιση θα περιλαμβάνει ζεύγη γνωρισμάτων που έχουν ταξινομηθεί κάτω από τις ίδιες κατηγορίες γνωρισμάτων. Στη συγκεκριμένη περίπτωση, όλα τα γνωρίσματα της CurrentSituation μπορούν να αντιστοιχιστούν με όλα τα γνωρίσματα της TracedSit1, γιατί ανήκουν στην κατηγορία γνωρισμάτων *dfdEntities* Αυτό σημαίνει ότι έχουμε 120 δυνατές αντιστοιχίσεις.



Σχήμα 4.1: Παράδειγμα εντοπισμού αναλογιών μεταξύ του TracedSit1 και του CurrentSituation

Συγκεκριμένα, σύμφωνα με το κριτήριο της σημασιολογικής ομοιογέ-

νειας, το γνώρισμα *CurrentSituation.dfd* , μπορεί να αντιστοιχιστεί τόσο στο *Tracesit1.resource* όσο και στο *Tracesit1.dfd* , αφού και τα δύο είναι περιπτώσεις της κατηγορίας γνωρισμάτων *dfdEntities* της κλάσης *PartiallyDescribedDFD*, όπως και το *CurrentSituation.dfd*.

Ο υπολογισμός των αποστάσεων μεταξύ των γνωρισμάτων απαιτεί τον υπολογισμό των ολικών αποστάσεων μεταξύ των τιμών των γνωρισμάτων αυτών. Το κριτήριο ελάχιστης απόστασης, αποτρέπει τις αντιστοιχίσεις γνωρισμάτων που οι τιμές τους έχουν μεγαλύτερη (ολική) εννοιολογική απόσταση από άλλες.

Έτσι, τελικά στη βέλτιστη αντιστοίχιση, το γνώρισμα *CurrentSituation.dfd* αντιστοιχεί στο *Tracesit1.dfd* αντί στο *Tracesit1.resource* , γιατί η εννοιολογική απόσταση (RentVehicle,CheckOutCopyOfBook) είναι μεγαλύτερη από την απόσταση (Vehicle,CheckOutCopyOfBook). Πιο συγκεκριμένα, η απόσταση του CheckOutCopyOfBook από το RentVehicle είναι η μικρότερη από τις αποστάσεις του CheckOutCopyOfBook με οποιοδήποτε από τα γνωρίσματα της κατηγορίας dfdEntities του Tracesit1.

Αυτό έγινε γιατί οι τιμές των γνωρισμάτων αυτών, το CheckOutCopyOfBook και το RentVehicle αντίστοιχα είναι εξειδικεύσεις της κλάσης *ResourceCheckOut* και είναι ταξινομημένα κάτω από την κλάση *BorrowingTransactionClass* Ενώ το Vehicle δεν έχει κοινή κλάση με το CheckOutCopyOfBook.

Στο σχήμα 4.1 βλέπουμε την βέλτιστη αντιστοίχιση των γνωρισμάτων, τη συνολική απόσταση των δύο συγκρινόμενων οντοτήτων καθώς και τις ολικές αποστάσεις των τιμών όλων των γνωρισμάτων που αντιστοιχήθηκαν. Στο κάτω μέρος του σχήματος βλέπουμε τον ορισμό σε Telos των περιστάσεων ως συγκροτήσεων που αποτελούνται απο στοιχεία του προϊόντος.

Κεφάλαιο 5

Καθοδήγηση της διαδικασίας παραγωγής προδιαγραφών: Ένα παράδειγμα

Θα περιγράψουμε τώρα ένα παράδειγμα που δείχνει με ποιο τρόπο εννοούμε την καθοδήγηση σ'αυτή την εργασία και πώς αυτή επιτυγχάνεται με την μέθοδο που προτείνουμε.

Το παράδειγμα που θα εξετάσουμε αφορά την διαδικασία δημιουργίας διαγραμμάτων ροής δεδομένων (data flow diagrams και για συντομία DFDs), τα οποία είναι πολύ συνηθισμένος τρόπος περιγραφής μοντέλων ενός συστήματος στη φάση της ανάλυσης απαιτήσεων. Τα DFDs χρησιμοποιούνται ευρύτατα στη μέθοδο της δομημένης ανάλυσης (SA - Structured Analysis[DeM79])

5.1 Το μοντέλο της διαδικασίας

Υποθέτουμε ότι η μέθοδος που σκοπεύουμε να ακολουθήσουμε, και περιγράφεται με ένα μοντέλο διαδικασίας, λέει πως για να περιγραφεί ένα απλό DFD πρέπει να καθοριστούν οι επεξεργασίες (processes), και οι ροές δεδομένων (data flows) που αποτελούν εισόδους και εξόδους κάθε επεξεργασίας, οι πηγές και καταβόθρες των δεδομένων, οι αποθήκες δεδομένων (data stores) από τις οποίες διαβάζει ή γράφει η διαδικασία(δηλαδή τους πόρους που χρησιμοποιούνται), ο πράκτορας που την ενεργοποιεί (triggering agent), αυτός που την εκτελεί (executing Agent), ο κάτοχος (possessing agent) του πόρου που αλλάζει κατάσταση, και η μετάβαση κατάστασης στην περίπτωση που υπάρχει αλλαγή κατάστασης.

Εστιάζουμε τώρα (στο θεωρούμενο μοντέλο διαδικασίας) στην περιγραφή του "input flow" κάποιας επεξεργασίας ενός απλού DFD που έχει ήδη μερικώς

περιγραφεί.

Η μέθοδος στην περίπτωση αυτή (όπου δηλαδή α. η πρόθεση μας είναι να περιγράψουμε ένα "bubble" του διαγράμματος, και β. η περίσταση στην οποία βρισκόμαστε περιγράφεται από τη μερική τυπική περιγραφή αυτού του bubble) προτείνει δύο τρόπους :
α.) να χρησιμοποιηθούν ήδη μοντελοποιημένες έννοιες - οντότητες (ροές)
β.) να δημιουργηθούν νέες.

Έστω ότι έχουμε επιλέξει το α.), δηλαδή, θέλουμε να χρησιμοποιήσουμε κάποια από τις ήδη υπάρχουσες ροές. Τίθεται λοιπόν το πρόβλημα του πώς να επιλέξουμε κάποια από αυτές. Οι επιλογές που δίνει το μοντέλο (βλ. σχήμα 5.1) από το σημείο αυτό της διαδικασίας, είναι οι εξής:

Να επιλέξουμε να περιγράψουμε το input χρησιμοποιώντας
1. τον πράκτορα που την εκτελεί (Executing Agent)
2. τον κάτοχο του πόρου (Possessing Agent)
3. τον πόρο που υφίσταται μετάβαση κατάστασης (Resource)
4. και τον κάτοχο του πόρου και τον ίδιο τον πόρο

Στο σχήμα βλέπουμε τη διάταξη των περιβαλλόντων απόφασης που αποτελούν το μοντέλο της διαδικασίας μας, με μορφή δέντρου. Κάθε κόμβος στο δέντρο αυτό καθορίζεται από ένα ζεύγος <περίσταση, απόφαση> και τα παιδιά του είναι οι επιλογές που έχουμε στη δεδομένη περίσταση, εφόσον πρόθεσή μας είναι να υλοποιήσουμε τη συγκεκριμένη απόφαση.

Οι ακμές είναι δύο ειδών:

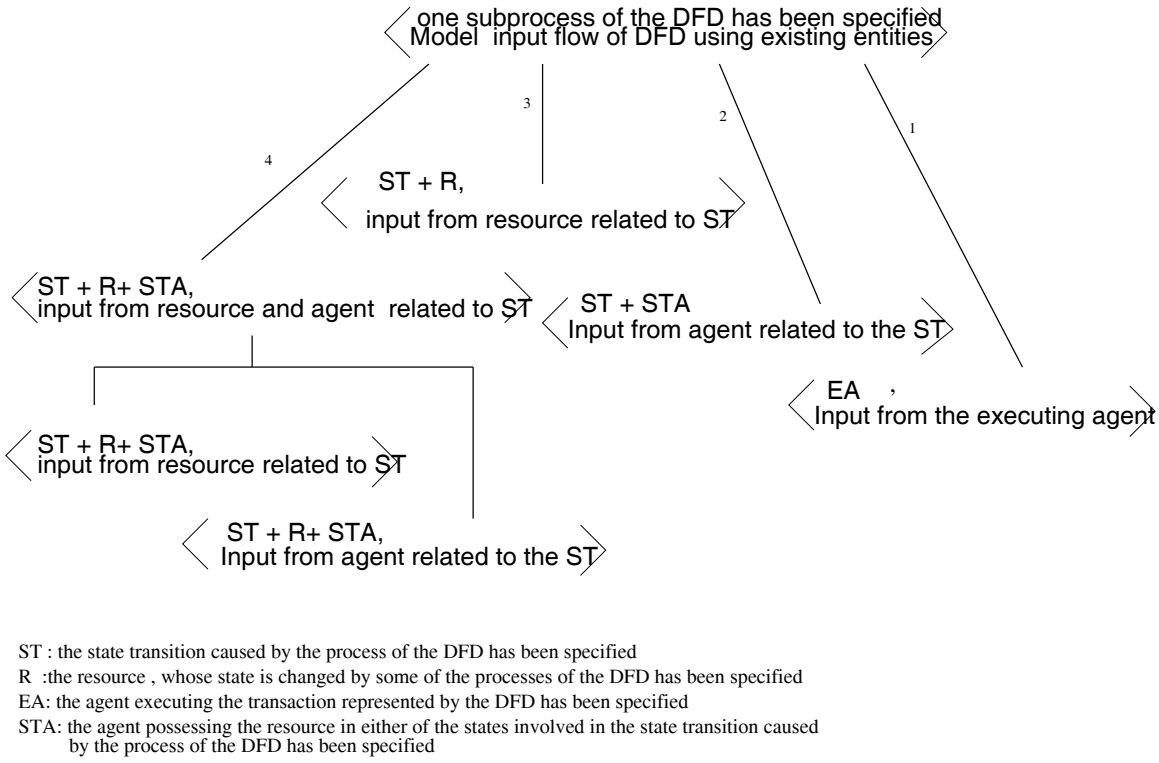
ακμές επιλογής , όπου για να προχωρήσουμε πρέπει να επιλέξουμε μία από αυτές (όπως οι ακμές που ξεκινάνε από τη ρίζα του δέντρου στο σχήμα 5.1)

και *ακμές ακολουθίας* , όπου για να προχωρήσουμε πρέπει να "περάσουμε" από όλες (όπως οι ακμές που ξεκινάνε από την επιλογή 4 στο σχήμα 5.1)

Οι καταστάσεις, που αποτελούν συγκροτήσεις στοιχείων του υπό κατασκευή προϊόντος, λειτουργούν σαν κάποια πρώτα φίλτρα επιλογής της θέσης μας στο δέντρο. Για παράδειγμα, βλέπουμε ότι για να είμαστε σε θέση να επιλέξουμε την περίπτωση 4, πρέπει η περίσταση να είναι [ST + R +STA] δηλαδή, στην περίσταση να περιέχονται ένα State Transition (αυτό που προκαλεί η περιγραφόμενη διεργασία) ένα Resource (αυτό που παθαίνει μεταβολή κατάστασης) και ο Agent (που σχετίζεται με την τελική κατάσταση στο State Transition).

Στο σχήμα βλέπουμε ακόμη ότι για να πραγματοποιήσουμε την απόφαση 4 (να ορίσουμε το input και από τον κάτοχο του πόρου και από τον ίδιο τον πόρο) πρέπει να εκτελέσουμε ακόμη δύο βήματα (και δε μας ενδιαφέρει η σχετική τους σειρά). Καθένα από αυτά αφορά την περιγραφή της κάθε συνιστώσας του input:

Να επιλέξουμε δηλαδή τον πόρο που υφίσταται μετάβαση κατάστασης και τον κάτοχό του σε κάποια από τις δύο καταστάσεις.



Σχήμα 5.1: Τμήμα του δέντρου διαδικασίας για την κατασκευή DFD

5.2 Το πρόβλημα

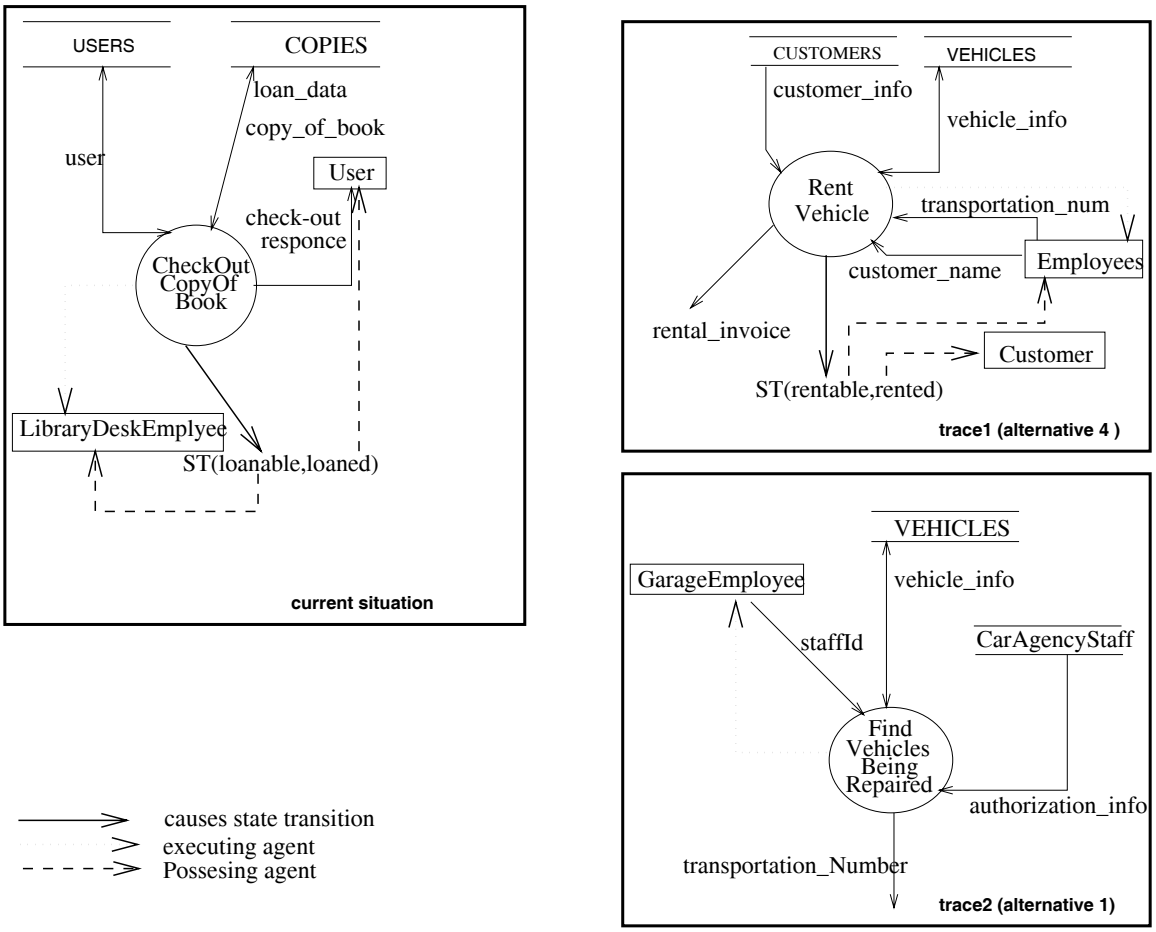
Θεωρούμε την εξής περίπτωση : Ακολουθώντας το μοντέλο διαδικασίας που περιγράφτηκε βρισκόμαστε στο σημείο που περιγράφουμε ένα DFD που αναφέρεται στο δανεισμό βιβλίου. Έχουμε ήδη περιγράψει κάποιες από τις οντότητες που σχετίζονται με το δανεισμό (LibraryBorrower, LibraryDeskEmployee, CopyOfBook) καθώς επίσης και το γεγονός ότι προκαλεί ένα state transition (ST(loanable, loaned)).

Στην εικόνα (σχήμα 5.2) αριστερά βλέπουμε σχηματικά την τρέχουσα περίσταση, δηλαδή το υπό κατασκευή DFD, που η θέση του στη διαδικασία είναι περίπτωση του *Model input flow of DFD using existing entities*, και παρατηρούμε ότι ικανοποιούνται όλες οι επιμέρους συνθήκες για όλες τις εναλλακτικές επιλογές. Είμαστε δηλαδή σε μια κατάσταση μεθοδολογικού διλήμματος και δεν εκφράζονται κριτήρια για τον αποκλεισμό κάποιας από τις εναλλακτικές

λύσεις.

5.3 Οι αποθηκευμένες περιπτώσεις (traces)

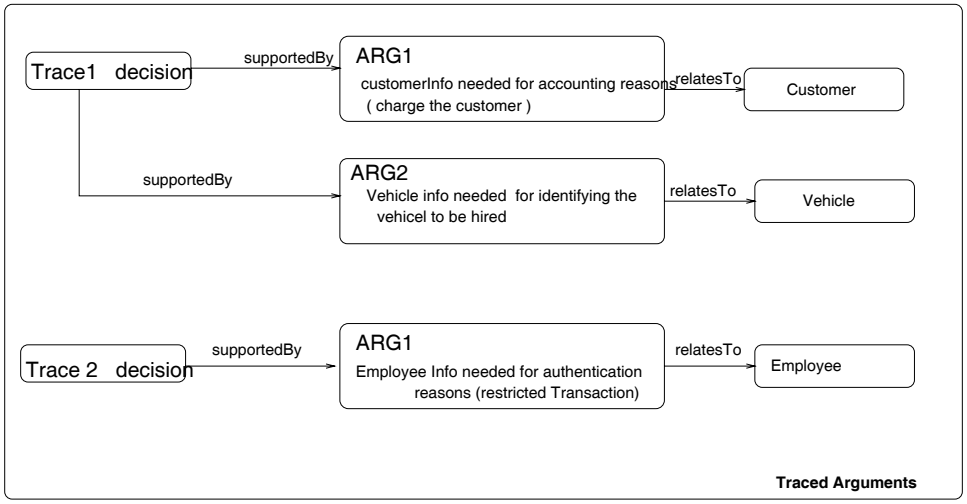
Έστω τώρα ότι στο repository μας υπάρχουν συγκεκριμένες περιπτώσεις από προηγούμενες "εκτελέσεις" του ίδιου μοντέλου διαδικασίας. Αυτές τις περιπτώσεις εκτελέσεων ενός μοντέλου διαδικασίας ονομάζουμε *ίχνη* (traces) της διαδικασίας.



Σχήμα 5.2: situations of current and traced examples

Στο σχήμα 5.2 (δεξιά) βλέπουμε και τις καταστάσεις από δυο συγκεκριμένες περιπτώσεις εκτέλεσης της ίδιας διαδικασίας, όπου έχουν ακολουθηθεί διαφορετικοί εναλλακτικοί τρόποι. Συγκεκριμένα, στο ένα έχουμε ένα DFD όπου έχει περιγραφεί η ενοικίαση αυτοκινήτου και είναι περίπτωση όπου έχουμε επιλέξει το 4 (δηλαδή σαν είσοδο στη δοσοληψία επιλέξαμε να δίνουμε στοιχεία και για τον πόρο που ψάχνουμε, αλλά και για τον πελάτη που ενεργοποιεί τη δοσοληψία), ενώ στο άλλο έχουμε το DFD της δοσοληψίας αναζήτησης των

αυτοκινήτων που βρίσκονται σε συνεργείο όπου έχουμε επιλέξει το 1(δηλαδή σαν είσοδο στη δοσοληψία επιλέξαμε να δίνουμε τα στοιχεία ταυτοποίησης του πράκτορα που την εκτελεί).



Σχήμα 5.3: Επιχειρήματα

Επιπλέον, για κάθε επιλογή εναλλακτικής λύσης που έχει γίνει έχουν καταχωρηθεί επιχειρήματα, έτσι ώστε να σχετίζονται με στοιχεία της περίστασης η οποία έθετε το δίλημμα. Τα συγκεκριμένα επιχειρήματα και οι συσχετίσεις τους με τμήματα της εκάστοτε περίστασης, φαίνονται στο σχήμα 5.3

Στο ίχνος 1, όπου η απόφαση είναι *"input from resource and Agent related to ST "* τα υποστηρικτικά επιχειρήματα ήταν ότι

α. χρειαζόμαστε στοιχεία από τον **πελάτη** για να μπορούμε να του χρεώσουμε το δανεισμό

και β. χρειαζόμαστε πληροφορία για το προς δανεισμό **όχημα** για να μπορέσουμε να το εντοπίσουμε

Στο ίχνος 2 η απόφαση είναι να δίνουμε σαν input στη δοσοληψία τον πράκτορα που την εκτελεί. Το υποστηρικτικό επιχείρημα εδώ είναι ότι χρειάζονται τα στοιχεία του γιατί πρέπει να επιβεβαιωθεί ότι ο συγκεκριμένος **υπάλληλος** έχει δικαίωμα να την εκτελέσει, μια και πρόκειται για δοσοληψία περιορισμένης πρόσβασης (authentication- restricted transaction).

5.4 Η λύση

Συγκρίνουμε (μέσω του SIS/ similarity analyzer[Spa94]) την τρέχουσα περίσταση με κάθε μία από τις δύο αποθηκευμένες καταστάσεις - περιπτώσεις εκτέλεσης.

Οι αναλογίες μεταξύ της τρέχουσας και της περίστασης του ίχνους που παράχθηκε κατά την κατασκευή του *RentVehicle* είναι οι εξής:

οντότητα τρέχουσας περίστασης	αντίστοιχη οντότητα ίχνους 1	κατηγορία γνωρίσματος
LibraryDeskEmployee	*Employee*	*Executing agent*
checkOutResponce	*VehicleInfo*	*output*
COPIES	*VEHICLES*	*readDataStores*
StateTransition(loanable,loaned)	*StateTransition(rentable,rented)*	std

Επίσης, μεταξύ των καταστάσεων CheckOutCopyOfBook (τρέχουσα) και FindVehiclesBeingRepaired (ίχνος 2) οι αναλογίες που βρίσκουμε είναι:

οντότητα τρέχουσας περίστασης	αντίστοιχη οντότητα ίχνους 2	κατηγορία γνωρίσματος
LibraryDeskEmployee	*GarageEmployee*	*executing agent*
CheckOutResponce	*TransportationNumber*	*output*
COPIES	*VEHICLES*	*readDataStores*

Παρατηρούμε ότι όλα τα στοιχεία της περίστασης του ίχνους που σχετίζονται με τα υποστηρικτικά επιχειρήματα της εναλλακτικής λύσης 4 έχουν ανάλογα (αντίστοιχα) στοιχεία στην περίσταση του υπό κατασκευή DFD. Παρατηρούμε ότι το επιχείρημα που υποστηρίζει την απόφαση 1 (μοντελοποίησης του input από τον Executing agent) στο ίχνος 2 σχετίζεται με το ένα μόνο από τα δύο ανάλογα στοιχεία των CheckOutcopyOfBook και FindVehiclesBeingRepaired.

Επειδή η τρέχουσα περίσταση παρουσιάζει περισσότερες ομοιότητες με την περίσταση του ίχνους 1 και τα ανάλογα στοιχεία σχετίζονται όλα με τα επιχειρήματα που στηρίζουν την απόφαση που πάρθηκε εκεί μπορούμε να υποθέσουμε ότι αυτός ο δεύτερος εναλλακτικός τρόπος συνιστάται ισχυρότερα.

Σε καμμιά περίπτωση δεν φιλοδοξούμε να υποδείξουμε τη μια και μοναδική σωστή λύση - κάτι τέτοιο δεν υφίσταται [Boo93]. Μπορούμε όμως ίσως να αναγνωρίσουμε γενικές περιπτώσεις όπου εφαρμόζονται κάποια heuristics - τρόπον τινά.

Αν περιορίσουμε τον υπολογισμό των αναλογιών σε εκείνα τα ίχνη που οι καταστάσεις τους εμφανίζουν ένα επιθυμητό βαθμό ομοιότητας με την τρέχουσα περίσταση, μπορούμε χωρίς να χρειαστεί να μελετήσουμε την μοντελοποίηση όλων των DFDs που πιθανόν να υπήρχαν στο repository μας να αποφασίσουμε ποια εναλλακτική λύση θα ακολουθήσουμε.

5.5 Μέθοδος

Η σύγκριση του τρέχοντος βήματος της διαδικασίας με τα αποθηκευμένα ίχνη, γίνεται ως εξής :

1. συγκρίνουμε το τρέχον βήμα με τα αποθηκευμένα ίχνη (όχι όλα, αλλά μόνο εκείνα που αποτελούν περιπτώσεις του ίδιου περιβάλλοντος απόφασης)

2. Υπολογίζουμε την ομοιότητα των καταστάσεων, οπότε προκύπτουν αναλογίες μεταξύ των στοιχείων που αποτελούν τις καταστάσεις τα οποία είναι στοιχεία από το προϊόν.

3.Μελετώντας τα ανάλογα στοιχεία από το προϊόν, βλέπουμε ποια από τα στοιχεία προϊόντος που περιγράφουν την εξελισσόμενη περίσταση έχουν ανάλογα στην περίσταση του ίχνους που επιπλέον συσχετίζονται με επιχειρήματα που στηρίζουν ή αντιτίθενται στην επιλογή της εναλλακτικής λύσης που έχει ακολουθηθεί στο ίχνος

4. μπορούμε να εκφράσουμε κάποια στοιχειώδη κριτήρια για το πως συνεχίζουμε : θεωρούμε σαν *καλύτερη προτεινόμενη* εναλλακτική αυτή για την οποία βρίσκουμε θετικά επιχειρήματα που βασίζονται σε ανάλογα product elements. Μπορούμε επίσης να θεωρούμε σαν *καλύτερη προτεινόμενη* εναλλακτική αυτή για την οποία υπάρχουν αρνητικά επιχειρήματα που σχετίζονται με στοιχεία κατάστασης του ίχνους που δεν έχουν ανάλογα στην τρέχουσα περίσταση.

- αυτή η λύση θα επιλεγεί αφού ο σχεδιαστής εκτιμήσει ότι οι αναλογίες περιλαμβάνουν σημαντικά στοιχεία της τρέχουσας περίσταση.

5.6 Συμπεράσματα

Σ'αυτό το παράδειγμα είδαμε μια εφαρμογή της μεθόδου μας σε ένα μικρό κομμάτι διαδικασίας- την κατασκευή DFDs. Το παράδειγμα αυτό είναι ενδεικτικό του πώς μπορεί να δοθεί καθοδήγηση όταν υπάρχουν αποθηκευμένα ίχνη μιας διαδικασίας και επιπλέον υπάρχει καταγεγραμμένο το σκεπτικό της κάθε απόφασης και μάλιστα με - έστω στοιχειώδη - δομημένο τρόπο (θετικά και αρνητικά επιχειρήματα).

Η τεκμηρίωση όμως των σχεδιαστικών αποφάσεων είναι κάτι που, λόγω του επιπλέον φόρτου εργασίας που εισάγει, δεν γίνεται κατά κανόνα. Κάτι τέτοιο όμως είναι απαραίτητο για να καταστεί αναχρησιμοποιήσιμη η ιστορία της διαδικασίας σχεδίασης. Γι'αυτό, θα επιχειρηθεί αργότερα να αντληθεί η σχετική πληροφορία με αναδιάθρωση της υλοποίησης (Reverse Engineering).

Θα επιχειρήσουμε στη συνέχεια μια προσέγγιση του προβλήματος με ένα γενικότερο μοντέλο διαδικασίας προσπαθώντας, για το λόγο αυτό, να καθοδη-

γήσουμε μια κοινά αποδεκτή και ευρέως χρησιμοποιούμενη μεθοδολογία για object oriented analysis and design (τη μεθοδολογία του Booch).

Κεφάλαιο 6

Η προσέγγισή μας

Στην εργασία μας χρησιμοποιούμε γνωστούς φορμαλισμούς και παράσταση σε Telos του μοντέλου της διαδικασίας. Δεν προτείνουμε ένα νέο φορμαλισμό για μοντέλα διαδικασιών, μια και υπάρχουν ήδη αρκετοί στη βιβλιογραφία.

Καταρχήν έχουμε επιλέξει από τη βιβλιογραφία ένα μεταμοντέλο για διαδικασίες.Αφού περιγράψουμε σε Telos το μεταμοντέλο αυτό, στη συνέχεια φτιάχνουμε μοντέλο διαδικασίας για μια συγκεκριμένη μεθοδολογία, σαν περίπτωση του μεταμοντέλου μας.

Η μεθοδολογία την οποία επιλέξαμε, είναι η διαδικασία σχεδίασης του Booch, (Booch Object Oriented Design, και για συντομία BOOD). Επιλέχτηκε γιατί είναι μια από τις πιο ευρέως χρησιμοποιούμενες μεθοδολογίες σχεδίασης με οντοκεντρική φιλοσοφία. Πήραμε την απαραίτητη γνώση για τη μεθοδολογία του Booch προκειμένου να παραστήσουμε την BOOD σε Telos, από το βιβλίο του Booch [Boo93] . Για το μεταμοντέλο της διαδικασίας χρησιμοποιήσαμε βασικές αρχές από αυτό που έχει προταθεί στα πλαίσια του ευρωπαϊκού προγράμματος βασικής έρευνας NATURE [1] [ROL94].

Έχοντας ορίσει το μοντέλο της BOOD σε Telos σύμφωνα με το μεταμοντέλο μας, θεωρούμε ότι έχουμε εκτελέσεις αυτής της διαδικασίας. Οι εκτελέσεις προκύπτουν σαν περιπτώσεις των περιβαλλόντων απόφασης του μοντέλου, τα οποία σχηματίζουν ακολουθίες.

Το νέο σ'αυτή την εργασία είναι η υποστήριξη της ερμηνείας ενός μοντέλου διαδικασίας από ένα εργαλείο υπολογισμού ομοιότητας (τον Similarity Analyzer [Spa94]), πάνω από τη βάση γνώσης της Telos στην οποία φυλάσσονται περιγραφές του μοντέλου της διαδικασίας και των περιπτώσεων εκτέλεσής του. Η

[1]ESPRIT BRA 6353 - Novel Approaches to Theories Underlying Requirements Engineering. The project partners are: University of Aachen (RWTH-Aachen), Germany (Project Coordinator) City University, London, U.K., Universite Paris I (Sorbonne), Paris, France ICS-FORTH, University of Heraklion, Greece and SISU-ISE, Royal Technical University of Stockholm, Kista, Sweden

συγκεκριμένη υλοποίηση της Telos που χρησιμοποιήσαμε, αναπτύχθηκε από το Ινστιτούτο Πληροφορικής του I.T.E [Mar92, Ντα93, Cos93]

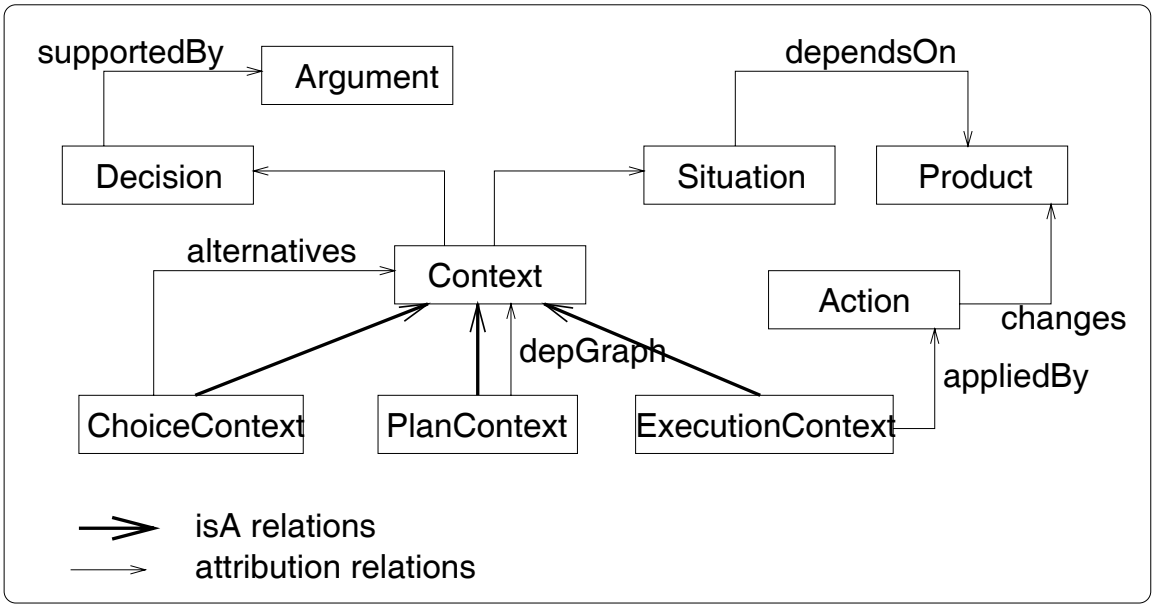
Πετυχαίνουμε, με τον παραπάνω συνδυασμό να ανακαλούμε ανάλογες περιπτώσεις εκτελέσεων, που συμβάλλουν στην ερμηνεία του μοντέλου της διαδικασίας χρησιμεύοντας σαν παραδείγματα σε περιπτώσεις μεθοδολογικών διλημμάτων.

6.1 Το μεταμοντέλο

Ένα μεταμοντέλο διαδικασιών είναι ένα μοντέλο δεδομένων ή σχήμα για την παράσταση σχεδιαστικών μεθόδων. Επιπλέον εκφράζει (και επιβάλλει) μια (κοινή) δομή εννοιών που είναι κοινές σε διάφορες μεθόδους.

Εδώ θα περιγράψουμε το μεταμοντέλο που χρησιμοποιήσαμε για να περιγράψουμε μοντέλα διαδικασιών. Οι οντότητες που αποτελούν ένα μοντέλο διαδικασίας, είναι περιπτώσεις των οντοτήτων που εισάγει το μεταμοντέλο. Κάθε τέτοιο μοντέλο περιγράφει μια μέθοδο, η οποία αποτελεί μοντέλο για πολλές διαδικασίες σχεδίασης και ανάλυσης που την ακολουθούν. Αυτές οι συγκεκριμένες διαδικασίες αποτελούν περιπτώσεις εκτέλεσης του μοντέλου διαδικασίας.

Το παρακάτω σχήμα (6.1) παριστάνει τις βασικές έννοιες του μεταμοντέλου μας και τις σχέσεις μεταξύ τους.



Σχήμα 6.1: Το μεταμοντέλο της διαδικασίας

Βλέπουμε ότι βασική έννοια του μεταμοντέλου είναι το **περιβάλλον απόφασης** (*Context*), Ένα περιβάλλον απόφασης συγκροτείται από την απόφαση που

παίρνουμε (*Decision*) (η οποία εκφράζει την πρόθεσή μας, ένα στόχο για να πραγματοποιηθεί) και την κατάσταση στην οποία βρίσκεται η διαδικασία, ή αλλιώς την **Περίσταση** (*Situation*). Η Περίσταση εξαρτάται από το **προϊόν**(*Product*). Για να καθοριστεί η περίσταση δεν απαιτείται ολόκληρο το προϊόν, αλλά τα τμήματα του εκείνα που επηρεάζουν ή σχετίζονται με την απόφαση. Επιπλέον, η απόφαση που περιλαμβάνεται σε ένα περιβάλλον απόφασης είναι δυνατόν να έχει και κάποια υποστηρικτικά επιχειρήματα (*Arguments*).

Το μεταμοντέλο διαδικασίας που χρησιμοποιούμε, λοιπόν, ανήκει στην κατηγορία των *decision oriented process models*, σε αντιδιαστολή με τα *product oriented* και τα *activity oriented* (βλ. σελ. 12), αφού σαν κεντρικό συστατικό του περιλαμβάνει την έννοια της απόφασης.

Το περιβάλλον απόφασης εξειδικεύεται σε

Περιβάλλον επιλογής (*ChoiceContext*) : χρησιμοποιείται για να εκφράσει την οριζόντια οργάνωση των δραστηριοτήτων. Περιγράφει σημεία επιλογής μεταξύ εναλλακτικών λύσεων όπου για να συνεχίσουμε μπορούμε να διαλέξουμε ΜΙΑ από τις εναλλακτικές λύσεις (*Alternatives*) που παρέχονται, οι οποίες είναι επίσης περιβάλλοντα απόφασης.

Περιβάλλον σχεδίου (*PlanContext*): χρησιμοποιείται για να εκφράσει την κάθετη οργάνωση των δραστηριοτήτων. Περιγράφει ακολουθίες από βήματα-περιβάλλοντα απόφασης (*Contexts*) που πρέπει να εκτελεστούν όλα τουλάχιστον μια φορά. Αυτά τα περιβάλλοντα απόφασης σχετίζονται με το περιβάλλον σχεδίου με το γνώρισμα (*depgraph*) Η σειρά με την οποία εκτελούνται καθορίζεται έμμεσα, από τη θέση (*Situation*) τους.

Περιβάλλον εκτέλεσης (*ExecutionContext*): εκφράζει πρόθεση να πραγματοποιηθούν ενέργειες (*Actions*) που επηρεάζουν άμεσα το προϊόν, προκαλώντας μετασχηματισμούς.Είναι το στοιχειώδες περιβάλλον απόφασης, και δεν αναλύεται περαιτέρω μέσα στο μεταμοντέλο.

Οι πράξεις εξειδικεύονται σε προσθετικές(CreateAction), μετατροπής(ChangeAction) ,και καταστροφής - σβησίματος (DeleteActions). Η επίδραση μιας πράξης στο προϊόν αποδίδεται με συσχετίσεις πράξεων με κατηγορίες του προϊόντος.

Το γεγονός ότι τα περιβάλλοντα σχεδίου και επιλογής αναλύονται, με συγκεκριμένο τρόπο, σε άλλα περιβάλλοντα απόφασης έχει σαν αποτέλεσμα να δημιουργείται μια μερική διάταξη, ή ιεραρχία, μεταξύ των περιβαλλόντων απόφασης στο επίπεδο του μοντέλου διαδικασίας, όταν δηλαδή θεωρούμε περιπτώσεις του μεταμοντέλου.

Μπορούμε να βλέπουμε ένα μοντέλο διαδικασίας σαν γράφο: Θεωρώντας τα περιβάλλοντα απόφασης σαν κόμβους και παριστάνοντας την ανάλυση ενός

περιβάλλοντος απόφασης σε βήματα ή επιλογές με ακμές που συνδέουν το PlanContext ή ChoiceContext αντίστοιχα με τα περιβάλλοντα στα οποία αναλύεται, ο γράφος που παριστάνει την ιεραρχία των περιβαλλόντων απόφασης, έχει τη μορφή δέντρου.Σ'αυτή θα αναφερόμαστε στο εξής σαν **δέντρο της διαδικασίας**.

Το μοντέλο της διαδικασίας, η μέθοδος δηλαδή, στη γενική περίπτωση είναι ένα δάσος από τέτοια δέντρα, και στην απλή περίπτωση όπου θεωρούμε ένα μόνο αρχικό περιβάλλον απόφασης (η διαδικασία ξεκινάει με τον ίδιο τρόπο) είναι ένα δέντρο.

Ο "αλγόριθμος διάσχισης" του δάσους της διαδικασίας είναι ο εξής

1. Επίλεξε από το δάσος ένα δέντρο, του οποίου η κορυφή έχει κενό τμήμα περίστασης (δηλαδή συνιστώσα περίστασης που δεν εξαρτάται από προϊόν), αφού ακόμη δεν υπάρχει προϊόν. Τρέχων κόμβος είναι η κορυφή του επιλεγμένου δέντρου

2. Επισκέψου τον τρέχοντα κόμβο:

   - αν είναι περιβάλλον εκτέλεσης, πραγματοποίησε την προτεινόμενη πράξη. Θέσε τρέχοντα κόμβο τον πατέρα, πήγαινε στο βήμα 2 (επισκέψου τον πατέρα)
   - αν είναι περιβάλλον επιλογής, διάλεξε ένα από τα παιδιά του ανάλογα με την τρέχουσα περίσταση. Αν από την περίσταση προκύπτει ότι είναι μοναδικό, επισκέψου το ·(τρέχων κόμβος το παιδί, πήγαινε στο βήμα 2) αλλιώς, κάλεσε *καθοδήγηση* [a] για την επιλογή του επόμενου τρέχοντος κόμβου.
   - αν είναι περιβάλλον σχεδίου, όσο (δεν έχεις επισκεφτεί όλα τα παιδιά του και θέλεις να συνεχίσεις) : διάλεξε ένα παιδί του (Π), ανάλογα με την τρέχουσα περίσταση (πήγαινε στο βήμα 2 με τρέχοντα κόμβο το Π) Αν έχεις επισκεφτεί μία φορά τουλάχιστον το κάθε παιδί, μπορείς να επιστρέψεις στον πατέρα του κόμβου. (πήγαινε στο βήμα 2 με τρέχοντα κόμβο τον πατέρα του κόμβου)

[a] η καθοδήγηση αφορά τη μέθοδο που περιγράφτηκε στο κεφάλαιο 4

Θα μπορούσε κανείς να πει ότι το δέντρο της διαδικασίας είναι ένα AND-OR tree, όπου οι AND κόμβοι είναι τα περιβάλλοντα σχεδίου και οι OR κόμβοι είναι

τα περιβάλλοντα επιλογής, με τη διαφορά ότι στους AND κόμβους μπορούμε να επισκεφτούμε τα παιδιά περισσότερες από μία φορές και επιπλέον πριν από κάθε επίσκεψη κόμβου ελέγχουμε αν μπορεί να εφαρμοστεί στην περίσταση.

6.2 Το μοντέλο της διαδικασίας BOOD

Θα περιγράψουμε τώρα το μοντέλο της διαδικασίας BOOD με βάση το μεταμοντέλο μας.

Κατ'αρχήν θα δούμε σε ποια σημεία της διαδικασίας ο σχεδιαστής καλείται να αποφασίσει, δηλαδή τις *θέσεις* των περιβαλλόντων απόφασης. Έπειτα θα δούμε τα περιβάλλοντα απόφασης που δομούνται πάνω σ'αυτές τις Θέσεις καθώς και το μοντέλο του προϊόντος, για την περιγραφή των προϊόντων της διαδικασίας που αποτελούν συστατικό των Θέσεων.

6.2.1 Περιστάσεις [περιβαλλόντων απόφασης](Situations)

Η μοντελοποίηση των περιστάσεων στις οποίες εφαρμόζεται μια απόφαση (Situations στο μεταμοντέλο) παρουσιάζει μεγάλο ενδιαφέρον για μας, γιατί αυτές αποτελούν τη βάση του αναλογικού συλλογισμού για την ανάκληση περιπτώσεων ανάλογων διλημμάτων [βλέπε κεφάλαιο 4]

Η Περίσταση, σαν συνιστώσα του περιβάλλοντος απόφασης, περιγράφει το πλαίσιο μέσα στο οποίο έχει νόημα η απόφαση αυτή. Παρόλο που η έννοια του περιβάλλοντος είναι δύσκολα οριοθετήσιμη,πρακτικά μπορούμε να την περιγράψουμε ικανοποιητικά (**??**) με βάση το μέχρι εκείνη τη στιγμή προϊόν της διαδικασίας. Για να αποδώσουμε τη γενικότητα του ρόλου που παίζει το προϊόν ως συνιστώσα περίστασης έχουμε επιλέξει να μοντελοποιήσουμε τη περίσταση σαν συγκρότηση τμημάτων του προϊόντος, και συγκεκριμένα σαν ένα σύνθετο αντικείμενο (aggregate object), που εισάγει μία κατηγορία γνωρισμάτων, το dependsOn (εξαρτάται από). Εφόσον η περίσταση εξαρτάται από το προϊόν, τα γνωρίσματα που ανήκουν στην κατηγορία αυτή (Situation.dependsOn), παίρνουν τιμές από κλάσεις και υποκλάσεις της μετακλάσης Product του μεταμοντέλου της διαδικασίας μας.

Στο μοντέλο μας της BOOD, οι τύποι των περιστάσεων (Situations) είναι οι εξής:

- *DefaultSituation*: Δηλώνει τη Περίσταση στην οποία το προϊόν μας δεν έχει αρχίσει ακόμη να κατασκευάζεται και αντιστοιχεί στην περίπτωση που ξεκινάμε να φτιάξουμε την περιγραφή του συστήματός μας με τη μέθοδο

BOOD Γενικότερα χρησιμεύει όταν κάποια απόφαση δεν εξαρτάται από κάποια συγκεκριμένα τμήματα του προϊόντος.

Για να εκφράσουμε το γεγονός ότι η κενή περίσταση (*DefaultSituation*) δηλώνει ότι μπορούμε να πάμε σ'αυτό το περιβάλλον απόφασης χωρίς περιορισμό περίστασης κάνουμε την *DefaultSituation* υποσύνολο όλων των τύπων περίστασης, ώστε οι περιπτώσεις κάθε τύπου περίστασης να μπορούν να είναι περιπτώσεις της.

- *ClassesAndObjects* ο γενικός τύπος περίστασης όπου έχουμε περιγράψει κάποιες από τις κλάσεις ή/και τα αντικείμενα του συστήματός μας · εξαρτάται από αντικείμενα τύπου Class και Object.

- *Objects* αυτός ο τύπος περίστασης δηλώνει ότι αρκεί να έχουμε ήδη ορίσει κάποια αντικείμενα, για να είναι ακολουθήσιμο το περιβάλλον απόφασης που καθορίζεται από αυτή την Περίσταση. Χρησιμοποιείται σε περιβάλλοντα απόφασης όπου η απόφαση αφορά κάποιον τύπο αντικειμένου ή ένα σύνολο αντικειμένων. Εξαρτάται από αντικείμενα τύπου Object.

- *Classes* αυτός ο τύπος περίστασης κατ'αναλογία με τον προηγούμενο, περιλαμβάνει τις καταστάσεις εκείνες όπου έχουν οριστεί κάποιες κλάσεις. Χρησιμοποιείται σε περιβάλλοντα απόφασης όπου η απόφαση αφορά κάποιον τύπο κλάσης ή ένα σύνολο κλάσεων. Εξαρτάται από αντικείμενα τύπου Class.

- *Operation*σ'αυτό τον τύπο περίστασης περιλαμβάνονται οι καταστάσεις εκείνες όπου έχουν ήδη οριστεί κάποιες λειτουργίες, χρησιμοποιείται σε περιβάλλοντα απόφασης όπου η απόφαση αφορά τις μεθόδους κάποιων κλάσεων.

- *RelationBetweenObjects*ανάλογα κι εδώ, σ'αυτό τον τύπο περίστασης περιλαμβάνονται οι καταστάσεις εκείνες όπου έχουν ήδη οριστεί κάποιες σχέσεις μεταξύ αντικειμένων και ειδικότερα σχέσης χρήσης. Εξαρτάται από αντικείμενα τύπου Object και τύπου Relation

- *UsageRelation* σ'αυτό τον τύπο περίστασης περιλαμβάνονται οι καταστάσεις εκείνες όπου έχουν ήδη οριστεί κάποιες σχέσεις χρήσης (τύπου Uses) μεταξύ κλάσεων. Εξαρτάται από αντικείμενα τύπου Object και τύπου Relation

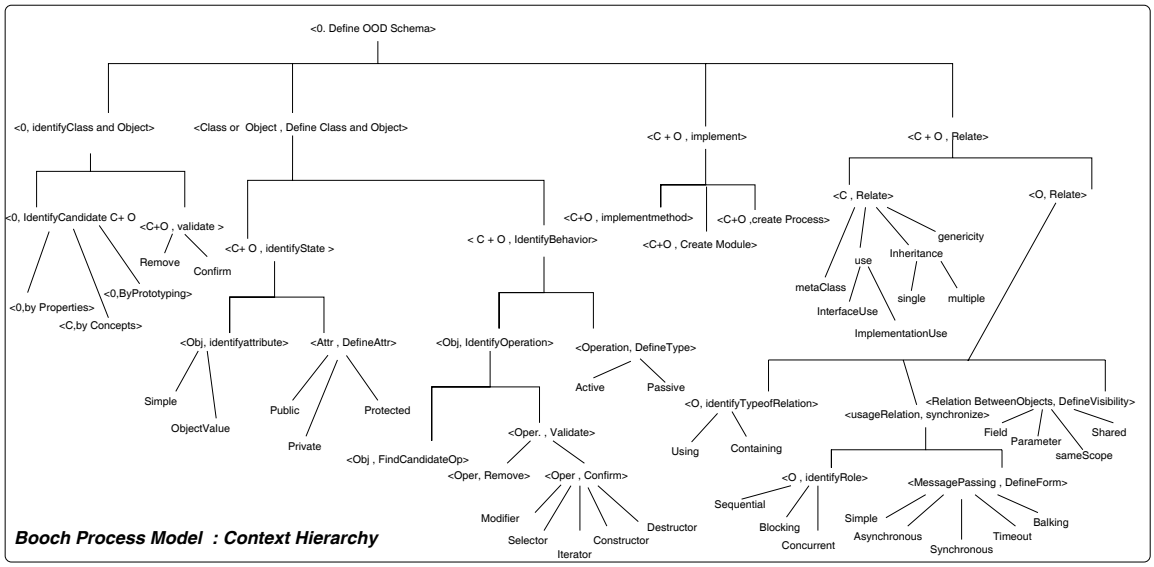
- *MessagePassing* Πρόκειται για ειδικότερη περίπτωση της περίστασης *UsageRelation* όπου η σχέση είναι τύπου περάσματος μηνυμάτων.

Αυτοί οι βασικοί τύποι καταστάσεων-θέσεων εμφανίζονται στο μοντέλο της διαδικασίας Booch. Για κάθε διαφορετική εκτέλεση της διαδικασίας δημιουργούμε περιπτώσεις αυτών των καταστάσεων με τα συγκεκριμένα προϊόντα από τα οποία εξαρτώνται κάθε φορά. Στις περιπτώσεις αυτές , εκμεταλλευόμαστε το γεγονός ότι στην Telos ένα γνώρισμα μπορεί να πάρει καμμία, μια ή περισσότερες τιμές, για να ομαδοποιήσουμε τις καταστάσεις που άλλοτε εξαρτώνται από μία π.χ. κλάση και άλλοτε αποτελούνται από ένα σύνολο κλάσεων.

6.2.2 Ανάλυση περιβαλλόντων απόφασης

Με βάση το μεταμοντέλο του σχήματος 6.1, τα περιβάλλοντα απόφασης της διαδικασίας του Booch περιγράφονται όπως φαίνεται στο παράρτημα (Telos source file: process_model.telos) Σχηματική παράσταση του δέντρου της διαδικασίας βλέπουμε στο σχήμα 6.2. Για να διακρίνουμε στο σχήμα τα περιβάλλοντα επιλογής από τα περιβάλλοντα που περιγράφουν ακολουθίες, παριστάνουμε διαφορετικά τις ακμές που τα συνδέουν με τα contexts στα οποία αναλύονται. Τα περιβάλλοντα εκτέλεσης (execution contexts) βρίσκονται στα φύλλα, αφού δεν αναλύονται περεταίρω σε άλλα περιβάλλοντα απόφασης.

Στη συνέχεια θα περιγράψουμε το μοντέλο της BOOD, αναλύοντας τα περιβάλλοντα απόφασης με τη σειρά που τα συναντούμε ακολουθώντας τον αλγόριθμο που έχουμε περιγράψει προηγουμένως.

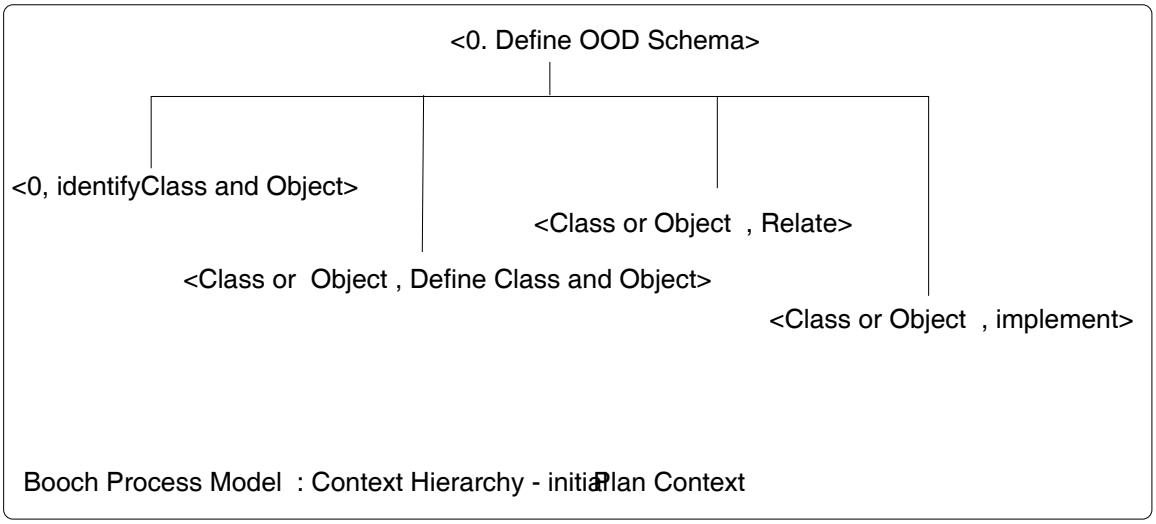


Σχήμα 6.2: Το μοντέλο της διαδικασίας BOOD

Η ρίζα του δέντρου στο σχήμα 6.2 παριστάνει το αρχικό περιβάλλον απόφασης. Αυτό είναι το ζεύγος < περίσταση, απόφαση> : **<0,DefineOOD Schema>**, που είναι ένα περιβάλλον σχεδίου.

Αναλυτικότερα , φαίνεται στο σχήμα 6.3.

Η κενή περίσταση σημαίνει ότι δεν έχουμε περιορισμούς περίστασης· και άρα μπορούμε να πάρουμε την απόφαση αυτή στην αρχή. Σ'αυτό το περιβάλλον απόφασης βλέπουμε ότι, σύμφωνα με τη μέθοδο του Booch, όταν θέλουμε να σχεδιάσουμε ένα σύστημα σύμφωνα με την OOD πρέπει να περάσουμε από τα τέσσερα περιβάλλοντα απόφασης στα οποία αναλύεται. Για τη σειρά με την οποία μπορούμε να προχωρήσουμε στην ακολουθία αυτή ισχύουν τα εξής:



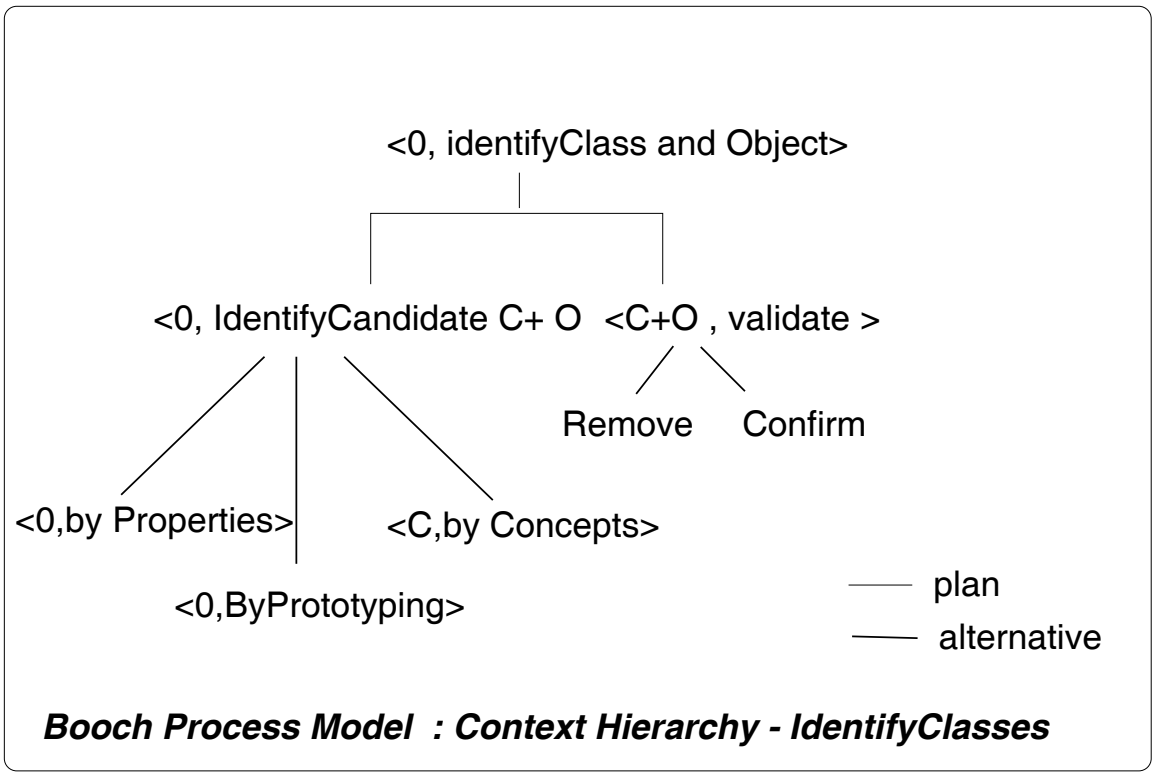
Σχήμα 6.3: Το αρχικό περιβάλλον απόφασης της BOOD

- Το πρώτο περιβάλλον απόφασης που θα ενεργοποιήσουμε είναι αυτό που δεν έχει περιορισμό περίστασης δηλαδή το **<0,Identify Classes and Objects>**. Είναι ένα περιβάλλον σχεδίου που αναλύεται στο **<0,IdentifyCandidate Classes and Objects>** και στο **<C+O,Validate>**. Για τα υπόλοιπα, μπορούμε να τα επισκεπτόμαστε με όποια σειρά και όσες φορές θέλουμε. Δηλαδή, μπορούμε να ακολουθούμε τα αντίστοιχα υποδέντρα σύμφωνα πάντα με τον αλγόριθμο που περιγράψαμε στη σελ. 44

  -- Κατ'αρχήν λόγω της κενής περίστασης, μπορούμε να εκτελέσουμε το **<0,IdentifyCandidate Classes and Objects>** το οποίο είναι ένα περιβάλλον επιλογής. Το υποδέντρο που έχει ρίζα αυτό το περιβάλλον απόφασης φαίνεται στο σχήμα 6.4

  Τα περιβάλλοντα επιλογής παριστάνονται με κόμβους που συνδέονται με τα παιδιά τους με ακμές που δεν τέμνονται. Το συγκεκριμένο, έχει παιδιά- επιλογές το **<0, By Properties>** , **<0, By Concepts>**, και **<0, By Prototyping >**. Αυτό σημαίνει ότι, όταν έχουμε κάποια Objects και θέλουμε να αναγνωρίσουμε αυτά που είναι υποψήφια να αποτελέσουν κλάσεις στο σύστημά μας, μπορούμε να το κάνουμε με τρεις τρόπους:

  * είτε κατατάσσοντάς τα σε σύνολα που έχουν χαρακτηριστικές ιδιότητες (**<0, By Properties >**). Πρόκειται για την κλασική μέθοδο της κατηγοριοποίησης γνωστή από τον Πλάτωνα, που όμως έχει το μειονέκτημα ότι δεν είναι δυνατό να κατασκευαστεί σύνολο ιδιοτήτων που να αρκεί για να καθορίζει τα όρια κάθε κατηγορίας, και άρα μπορεί σε κάποιες περιπτώσεις να αποτύχει



Σχήμα 6.4: Το δέντρο του εντοπισμού κλάσεων και αντικειμένων της BOOD

να εκφράσει τη δομή των κλάσεων όπως την έχει συλλάβει ο σχεδιαστής.

- είτε να τα κατατάξουμε με λιγότερο αυστηρά κριτήρια (**<0, By Concepts>**), με βάση ιδιότητες που δεν είναι μετρήσιμες, όπως π.χ. για την κλάση των χαρούμενων ανθρώπων. Αυτό γίνεται δημιουργώντας κατ'αρχήν εννοιολογικές περιγραφές των κατηγοριών και στη συνέχεια κατατάσσοντας τα αντικείμενα με βάση τις περιγραφές τους
- είτε μέσω συσχέτισης τους με κάποιο πρότυπο, (**<0, By Prototyping >**). Εδώ κατ'αρχήν θεωρούμε κάποιο αντικείμενο σαν το αντιπροσωπευτικό παράδειγμα της κλάσης. Στη συνέχεια κατατάσσουμε τα αντικείμενα σύμφωνα με το βαθμό συσχέτισης τους με το αντιπροσωπευτικό παράδειγμα της κλάσης (το πρότυπο).

Συνήθως ξεκινάμε την προσπάθεια επιλογής κλάσεων μέσω ιδιοτήτων που εντοπίζουμε μελετώντας τη δομή των αντικειμένων που αναφέρονται στο λεξικό του προβλήματος(**By Properties**). Αν δεν προκύψει έτσι ικανοποιητική δομή κλάσεων, εξετάζουμε τη συμπε-

ριφορά των αντικειμένων που αλληλεπιδρούν εντοπίζοντας ιδιότητες που αποδίδουν χαρακτηρισμούς (**By Concepts**), εδώ *επινοούμε* κλάσεις, δημιουργώντας κοινή υπερκλάση για κλάσεις που έχουν κοινές υπευθυνότητες · αν ούτε και τότε πετύχουμε να εκφράσουμε τη δομή κλάσεων όπως την έχουμε συλλάβει τότε προσπαθούμε **By Prototyping**

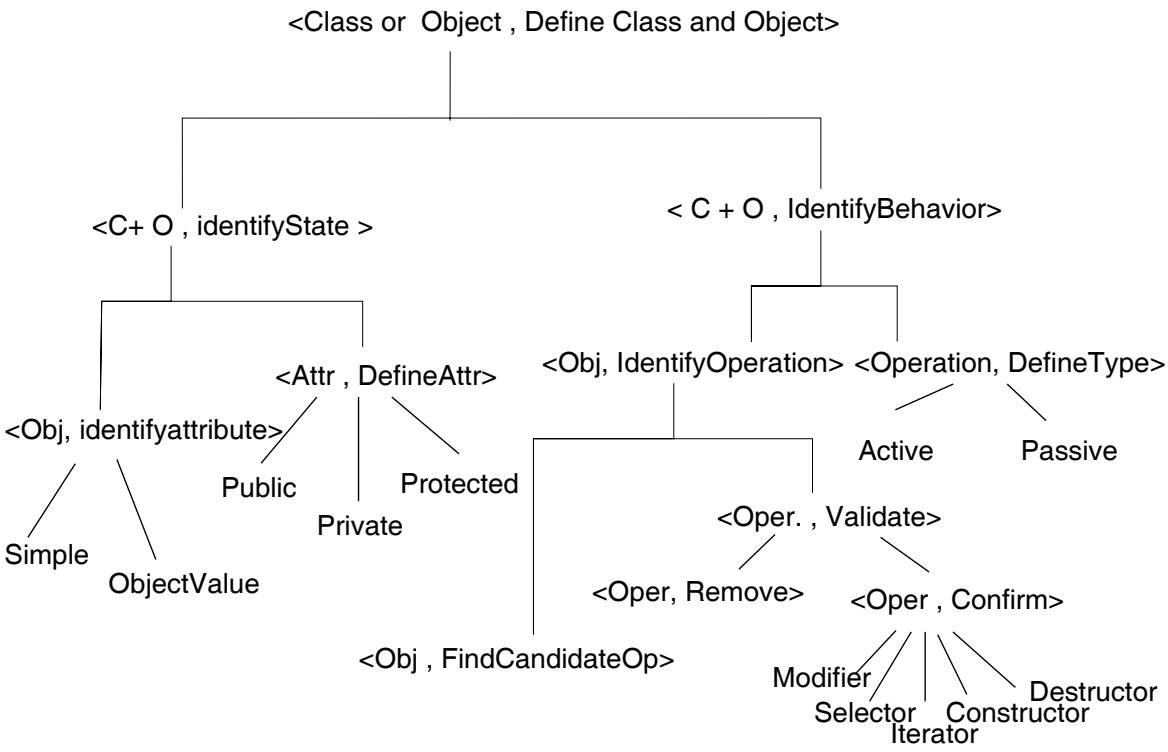
-- Στη συνέχεια, αφού έχουμε επιλέξει κάποιες υποψήφιες κλάσεις και αντικείμενα, πρέπει να επικυρώσουμε τις επιλογές μας. Αυτό γίνεται στο περιβάλλον επιλογής **<C+O, Validate>**. Η περίσταση **C+O** αποτελείται από κάποια από τα υποψήφια αντικείμενα ή / και κλάσεις. Εφόσον πρόκειται για περιβάλλον επιλογής, μπορούμε να επιλέξουμε για κάθε συγκεκριμένο αντικείμενο ή κλάση (που αναφέρεται στη Περίσταση C+O) να επιλέξουμε μεταξύ των εξής: είτε να επιβεβαιώσουμε την επιλογή της συγκεκριμένης κλάσης ή αντικείμενου (**< C or O, Confirm >**) είτε να την απορρίψουμε (**< C or O, Remove>**). Για να επιβεβαιώσουμε μια κλάση, πρέπει να θεωρούμε ότι ικανοποιεί τα εξής: όλες οι περιπτώσεις της κλάσης παρουσιάζουν κοινή συμπεριφορά, περιέχουν μια εσωτερική περίσταση και καθένα από αυτά έχει μοναδική ταυτότητα.

Σ'αυτό το περιβάλλον απόφασης, όπως και στο <0,IdentifyCandidate Classes and Objects> μπορούμε να επανερχόμαστε όσο έχουμε υποψήφιες κλάσεις και αντικείμενα ή όσο θέλουμε να ανακαλύπτουμε καινούρια.

Όταν επιλέξουμε να συνεχίσουμε, μπορούμε να προχωρήσουμε στη συσχέτιση (ορισμό σχέσεων) ή στον ορισμό ή στην υλοποίηση των επικυρωμένων κλάσεων ή αντικειμένων.

- Αν επιλέξουμε να ορίσουμε τις κλάσεις μας τότε η επόμενη κίνησή μας καθορίζεται από το περιβάλλον σχεδίου **<Class or Obj, Define Class or Obj>** Η ιεραρχία των περιβαλλόντων απόφασης του ορισμού κλάσεων φαίνεται στο σχήμα 6.5 Αυτό αναλύεται σε δύο περιβάλλοντα σχεδίου με την ίδια περίσταση (C or O) και αποφάσεις Identifystate και IdentifyBehavior αντίστοιχα. Για να ορίσουμε λοιπόν κάποια κλάση ή αντικείμενο πρέπει να καθορίσουμε τη δομή του (Identifystate) και τη συμπεριφορά του (IdentifyBehavior).

  -- Ο καθορισμός της δομής γίνεται επιλέγοντας το περιβάλλον σχεδίου **<C or O, identifyState>** που, αναλύεται στα **<Obj, IdentifyAttribute>** και **<Attr, DefineAttr>**. Δηλαδή με δεδομένο ένα αντικείμενο, καθορίζουμε υποψήφια γνωρίσματα.Αυτό σημαίνει ότι καθορίζουμε τον



Σχήμα 6.5: Το δέντρο του ορισμού κλάσεων και αντικειμένων της BOOD

τύπο τους είτε επιλέγοντας να είναι απλά γνωρίσματα, είτε αποδίδοντας κάποια αντικείμενα σαν γνωρίσματα άλλων, και τα ορίζουμε, καθορίζοντας την προσβασιμότητά τους (δημόσια - **public**, ιδιωτικά - **Private** ή προστατευόμενα - **Protected**)

-- Αφού το κάνουμε αυτό για όσα γνωρίσματα θέλουμε, μπορούμε να προχωρήσουμε στον καθορισμό της συμπεριφοράς των αντικειμένων.Σαν συμπεριφορά εννοείται ο τρόπος με τον οποίο ένα αντικείμενο ενεργεί και αντιδρά, σε σχέση με άλλα από την άποψη της αλλαγής κατάστασης και του περάσματος μηνυμάτων. Το αντίστοιχο περιβάλλον απόφασης, το **<C or O, identifyBehavior>**, είναι περιβάλλον σχεδίου. Σύμφωνα μ'αυτό, για να καθορίσουμε τη συμπεριφορά μιας κλάσης ή αντικειμένου πρέπει να ορίσουμε τις λειτουργίες που θα έχει και να ορίσουμε τον τύπο τους.

  * Για να ορίσουμε τις λειτουργίες (**<Obj,IdentifyOperation>**) πρέπει κατ'αρχήν να εντοπίσουμε υποψήφιες δηλαδή δράσεις ενός αντικειμένου πάνω σε άλλο με στόχο την πρόκληση αντίδρασης από

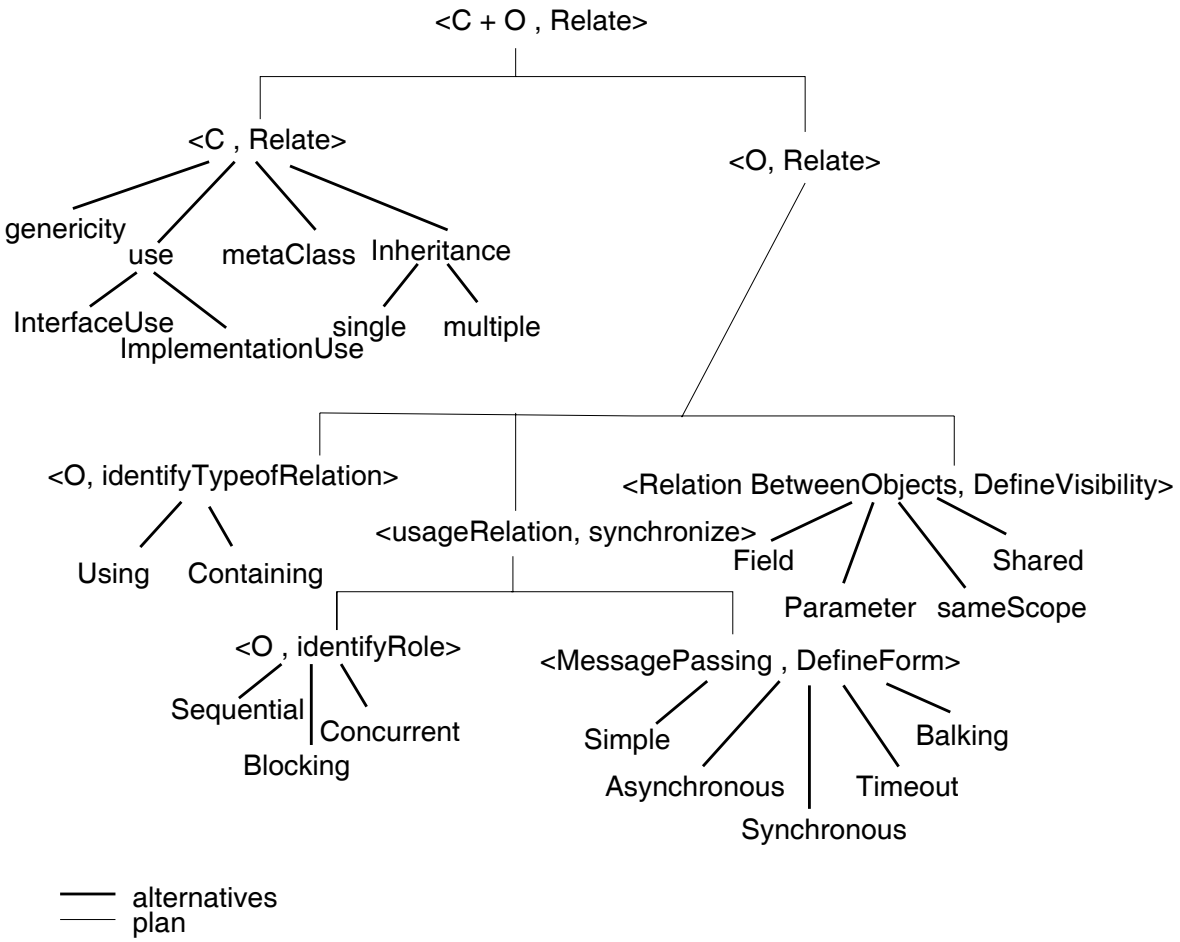
το δεύτερο και στη συνέχεια να τις επικυρώσουμε.

* Η επικύρωση (**<Operation, Validate>**) περιγράφεται από ένα περιβάλλον επιλογής. Οι επιλογές που δίνονται είναι είτε να σβήσουμε την επιλεγμένη κλάση, είτε να την επιβεβαιώσουμε.
* Η επιβεβαίωση περιγράφεται από ένα περιβάλλον επιλογής, το **< Operation, Confirm>** και για να γίνει πρέπει να καθορίσουμε αν η λειτουργία αυτή είναι
  - **Modifier** δηλαδή αλλάζει την κατάσταση (αλλάζει τιμή σε μεταβλητές) του αντικειμένου
  - ή επιλογέας **Selector** (έχει πρόσβαση στην τιμή κάποιας από τις μεταβλητές του αντικειμένου αλλά όχι δικαίωμα να την αλλάξει)
  - ή κατασκευαστής (**Constructor**) του αντικειμένου
  - ή ρουτίνα καταστροφής του αντικειμένου **(Destructor)**
  - ή απλά είναι μια λειτουργία που επιτρέπει πρόσβαση σε όλα τα τμήματα του αντικειμένου με κάποιον καλά ορισμένο τρόπο (**Iterator**)

- Αν επιλέξουμε να συσχετίσουμε κλάσεις και αντικείμενα, οι δυνατές πράξεις μας περιγράφονται από το περιβάλλον σχεδίου **<C+O, Relate>** το οποίο καθορίζει ότι πρέπει να συσχετίσουμε τις κλάσεις και να συσχετίσουμε τα αντικείμενα.Σχηματικά το υπόδεντρο της διαδικασίας φαίνεται στο σχήμα 6.6

  -- Για να συσχετίσουμε κλάσεις (περιγράφεται από το περιβάλλον επιλογής **<C,Relates>**) έχουμε τις επιλογές ανάλογα με τον τύπο της σχέσης:

  * να ορίσουμε σχέση περιέχεσθαι μεταξύ της μίας κλάσης και της άλλης, δηλαδή σχέση μετακλάσης.
  * να ορίσουμε σχέση χρήσης (**uses**) οπότε έχουμε επιπλέον επιλογή
    - η μία κλάση να χρησιμοποιεί την υλοποίηση της άλλης (**ImplementationUse**)
    - η μία κλάση να χρησιμοποιεί την επαφή χρήσης της άλλης **(InterfaceUse)**
  * να ορίσουμε σχέση γενίκευσης
  * να ορίσουμε σχέση κληρονόμησης όποτε πρέπει επιπλέον να επιλέξουμε αν θα έχουμε απλή (**SimpleInheritance**) ή πολλαπλή κληρονόμηση (**MultipleInheritance**).



Booch Process Model : Context Hierarchy - Relate Classes and Objects

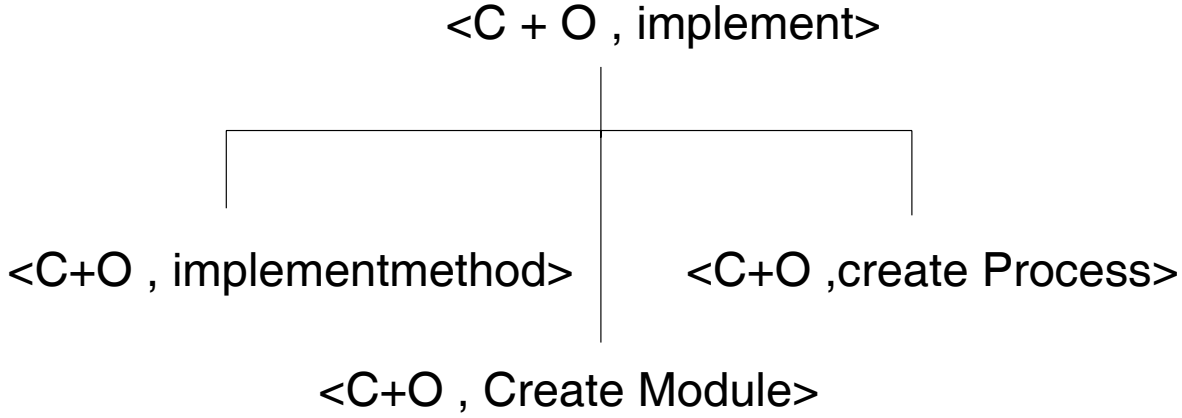
Σχήμα 6.6: Το δέντρο του συσχετισμού κλάσεων και αντικειμένων της BOOD

-- Για να συσχετίσουμε αντικείμενα(περιγράφεται από το περιβάλλον σχεδίου **<0,Relate>**) πρέπει να κάνουμε τα εξής:

* Με δεδομένο το αντικείμενο να ορίσουμε τον τύπο της συσχέτισης **<O, IdentifyTypeOfRelation>**, που είναι ένα περιβάλλον επιλογής με επιλογές να φτιάξουμε συσχέτιση τύπου χρήσης ή τύπου περιέχεσθαι.
* Με δεδομένο τη συσχέτιση, να καθορίσουμε τους περιορισμούς ορατότητας που ισχύουν, δηλαδή αν τα συσχετιζόμενα αντικείμενα "βλέπουν" το ένα το άλλο επειδή το ένα είναι πεδίο (**Field**) ή παράμετρος (**parameter**) στο άλλο, αν βρίσκονται στην ίδια εμβέλεια (**Same scope**), ή αν έχουν κοινή κατάσταση (**shared**)
* Με δεδομένη τη σχέση χρήσης μεταξύ δύο αντικειμένων,

να καθορίσουμε πώς συγχρονίζονται (περιβάλλον σχεδίου **<usageRelation,synchronize>**). Ουσιαστικά εδώ δίνουμε τιμή στο γνώρισμα Synchronization των αντικειμένων (βλέπε και παράγραφο που περιγράφονται τα διαγράμματα αντικειμένων). Στη συνέχεια πρέπει

- να καθορίσουμε το ρόλο του κάθε αντικειμένου, αν δηλαδή είναι συνδρομικό (**concurrent**), ακολουθιακό **(sequential)** ή **Blocking** δηλαδή θα ”κολάει περιμένοντας την απάντηση του άλλου)
- να καθορίσουμε,με δεδομένο ότι τα αντικείμενα επικοινωνούν με πέρασμα μηνυμάτων τη μορφή της επικοινωνίας, δηλαδή αν είναι απλή (**simple**), σύγχρονη (**synchronous**), ασύγχρονη **asynchronous**, **balking**, ή **timeout**.



Σχήμα 6.7: Το δέντρο της υλοποίησης της BOOD

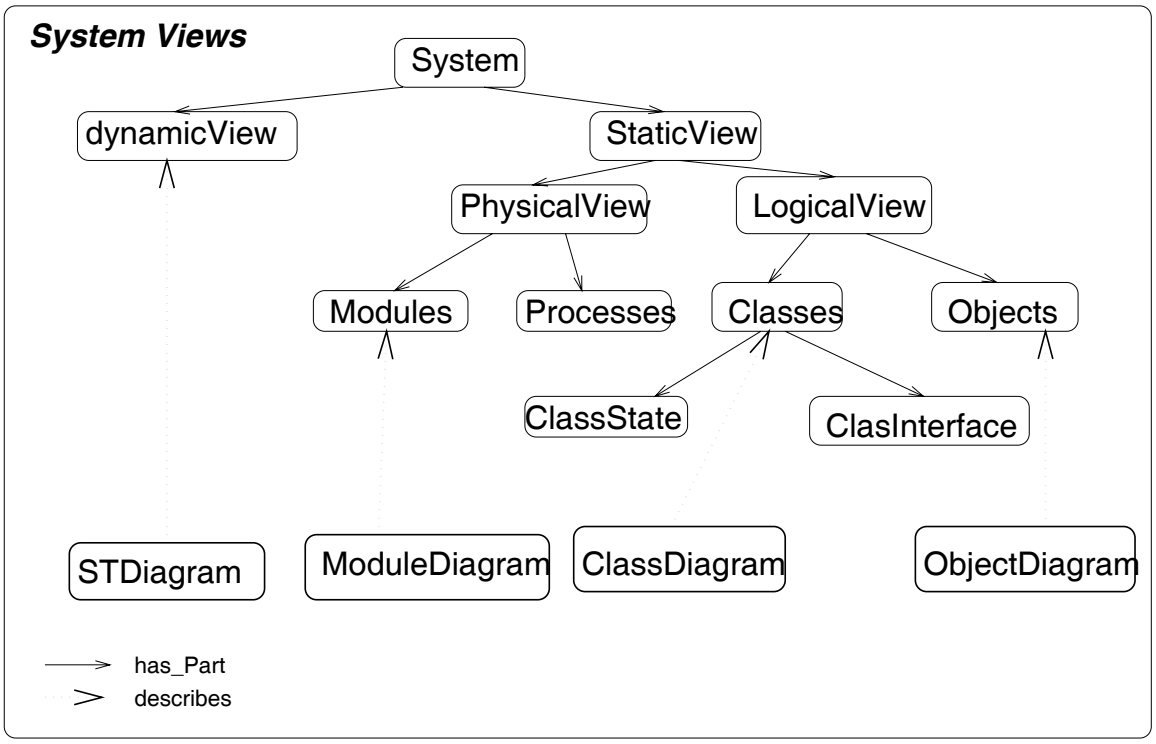
- Αν επιλέξουμε να υλοποιήσουμε κάποιες κλάσεις ή αντικείμενα, (περιβάλλον σχεδίου **<C+O, Implement>**) τότε πρέπει σύμφωνα με το μοντέλο (σχήμα 6.7) να προχωρήσουμε ως εξής:

  -- να ορίσουμε την υλοποίηση των μεθόδων τους (**< C + O, ImplementMethods>**)

  -- να χωρίσουμε τον κώδικα σε modules, δηλαδή να περιλάβουμε σε κοινά αρχεία τις υλοποιήσεις κλάσεων και μεθόδων με κάποιο καθορισμένο τρόπο (**< C+O, CreateModule>**)

-- να δημιουργήσουμε τις υλοποιήσεις των αντικειμένων που είναι ενεργά ή το κυρίως πρόγραμμα, καθορίζοντας έτσι τα νήματα ελέγχου (threads of control) του συστήματός μας (< **C+O, CreateProcesses**>)

6.2.3 Το μοντέλο του προϊόντος (Product Model)



Σχήμα 6.8: Όψεις συστήματος κατά Booch

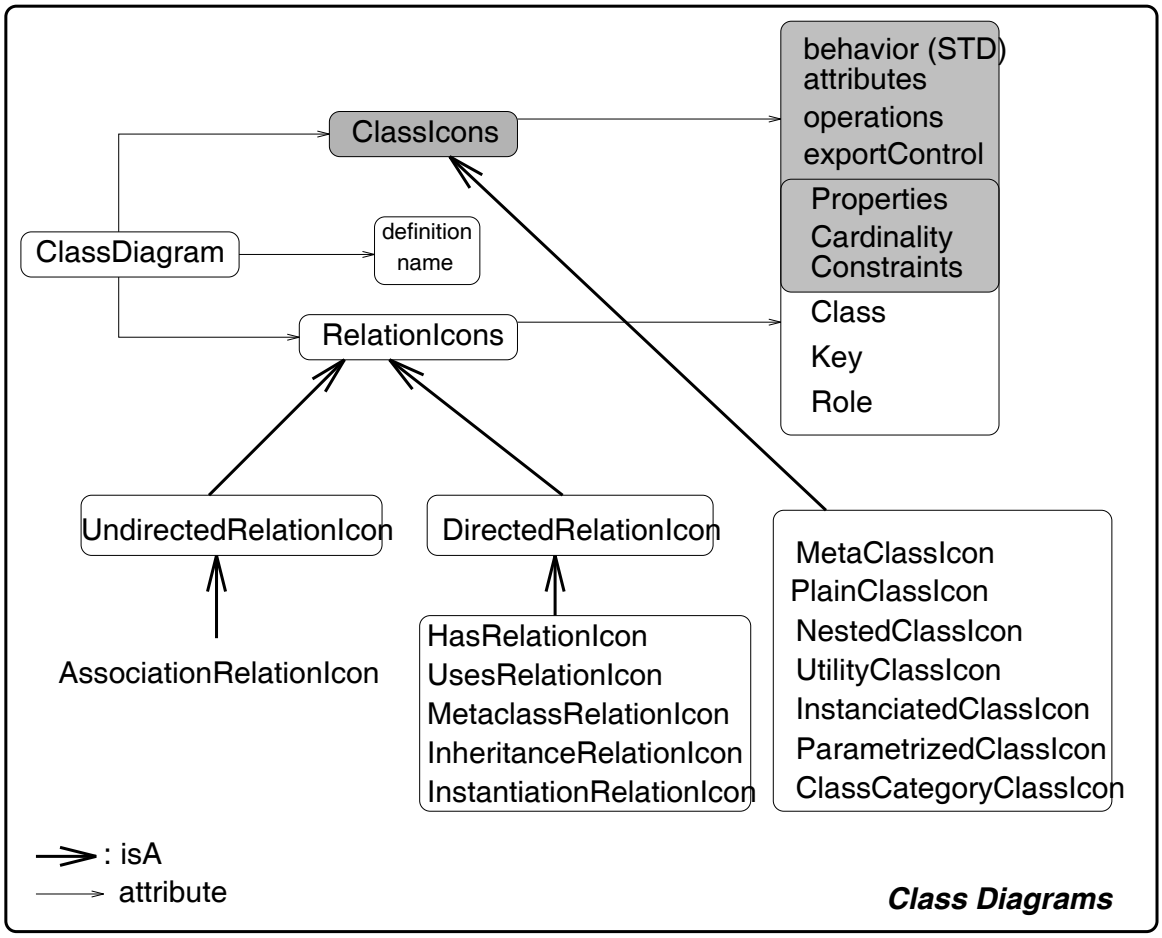
Ένα σύστημα για τον Booch παρουσιάζει διάφορες όψεις. που σχηματικά φαίνονται στο διάγραμμα του σχήματος 6.8

Κάθε όψη του συστήματος προδιαγράφεται με ένα είδος διαγράμματος. Αναλυτικά θα δούμε παρακάτω τα διάφορα μοντέλα διαγραμμάτων. Η κατασκευή αυτών των μοντέλων βασίστηκε στην ιδέα ότι ένα διάγραμμα περιέχει συσχετιζόμενα σύμβολα τα οποία παριστάνουν κάποιες έννοιες του μοντέλου του συστήματος.

Κατ'αρχήν εδώ μας απασχόλησε η παράσταση ενός διαγράμματος με τυπικό τρόπο. Στη συνέχεια και με δεδομένο το μοντέλο μας για τα διαγράμματα, η περιγραφή τους σε Telos μπορεί να παραχθεί αυτόματα από γραπτές περιγραφές σε κάποια τυπική γλώσσα περιγραφής διαγραμμάτων. Για την περιγραφή των διαγραμμάτων σε Telos, δημιουργήσαμε την κλάση Diagrams η οποία περιλαμβάνει όλα όσα παριστάνονται σε ένα διάγραμμα (DiagramElements).

Διαγράμματα κλάσεων

Ένα διάγραμμα κλάσεων παριστάνει βασικές αφαιρέσεις των κλάσεων και των διαφορετικών σχέσεων μεταξύ τους.



Σχήμα 6.9: Το μεταμοντέλο για τα διαγράμματα κλάσεων

Στη φάση της ανάλυσης, όπου σκιαγραφείται η συμπεριφορά του συστήματος, τα διαγράμματα κλάσεων χρησιμεύουν για να δείχνουν τους κοινούς ρόλους και τις ευθύνες που ανατίθενται στις διάφορες οντότητες. Στη φάση της σχεδίασης, όπου σκιαγραφείται η αρχιτεκτονική του συστήματος, χρησιμεύουν ως μέσο σύλληψης της δομής των κλάσεων.

Τα συστατικά στοιχεία ενός τέτοιου διαγράμματος είναι εικονίδια κλάσεων και σχέσεων.Στο μοντέλο μας αυτά παριστάνονται με τα αντικείμενα *ClassIcon* και *RelationIcon.* Κάθε εικονίδιο έχει ένα τύπο (ανάλογα με το αν παριστάνει σχέση ή αντικείμενο) και κάποιες χαρακτηριστικές ιδιότητες. Οι ιδιότητες αυτές στο διάγραμμα του Booch εμφανίζονται είτε σαν εκφράσεις με δεδομένο συντακτικό είτε σαν χαρακτηριστικά εικονίδια και στο μοντέλο μας αποδίδονται σαν γνωρίσματα των αντικειμένων *ClassIcon* και *RelationIcon.* Για μας όλες οι

ιδιότητες ενός εικονιδίου, ανεξάρτητα από το πως εμφανίζονται στο διάγραμμα, κατατάσσονται στην κατηγορία DiagramElements και σε υποκατηγορίες ανάλογα με το είδος του εικονιδίου.

Στο σχήμα 6.9 βλέπουμε (με όρους της Telos) το μοντέλο των διαγραμμάτων που παριστάνουν κλάσεις και τις μεταξύ τους σχέσεις.

Τα διαγράμματα κλάσεων στον Booch αντιμετωπίζονται σαν ατομικές οντότητες μέσα στη μακρο διαδικασία. Εμείς επειδή θέλουμε να εντοπίζουμε αναλογίες των συστατικών στοιχείων συγκεκριμένων διαγραμμάτων με στοιχεία άλλων διαγραμμάτων ή με τις τυπικές μορφές σε γλώσσα προγραμματισμού, στις οποίες τελικά εξελίχτηκαν, επιλέγουμε να τα παριστάνουμε σαν συγκροτήσεις των στοιχείων τους.

Μια εναλλακτική λύση για την παράσταση των διαγραμμάτων ήταν η παράστασή τους σαν "μαύρα κουτιά" που μας ενδιαφέρει η ύπαρξή τους μόνο και όχι το περιεχόμενό τους, (δηλαδή χρησιμοποιώντας δείκτες στο αρχείο που τα περιγράφει). Δεν προτιμήθηκε όμως για τους λόγους που προαναφέρθηκαν, παρόλο που είναι συμβατή με την ιδέα ότι το διάγραμμα σαν οντότητα αποτελεί στοιχείο περίστασης της μακρο- διαδικασίας

Οι κατηγορίες γνωρισμάτων που εισάγουν τα αντικείμενα ClassIcon και RelationIcon είναι :

- η κατηγορία *properties*, που αφορά ιδιότητες που αποδίδονται σε κλάσεις ή σε σχέσεις μεταξύ κλάσεων. Η απόδοση ιδιότητας σε κάθε συγκεκριμένο διάγραμμα γίνεται με τη χρήση ενός χαρακτηριστικού εικονιδίου. Οι περιπτώσεις της κατηγορίας αυτής είναι :

  -- **A**bstract, στα διαγράμματα παριστάνεται με **A** και χαρακτηρίζει αφηρημένες κλάσεις, δηλαδή κλάσεις που δεν έχουν άμεσες περιπτώσεις.

  -- **F**riend χρησιμεύει για να δείξει ότι οι κλάσεις που συνδέονται μεταξύ τους με σχέσεις που έχουν αυτή την ιδιότητα, δίνουν (ανάλογα με την κατεύθυνση της σχέσης) η μία στην άλλη δικαίωμα πρόσβασης στα μη δημόσια τμήματα τους.

  -- **V**irtual που χαρακτηρίζει σχέσεις κληρονόμησης και δείχνει ότι η κληρονόμηση είναι εικονική.Εμφανίζεται σε περιπτώσεις όπου η ιεραρχία ταξινόμησης σχηματίζει "ρόμβους", για να υποδηλώσει την κοινή βασική κλάση δύο εξειδικεύσεων από τις οποίες κληρονομεί μια τρίτη κλάση.

  -- **S**tatic, χαρακτηρίζει σχέσεις τύπου *has* και δηλώνει member objects ή functions, που ανήκουν στην κλάση αλλά όχι στις περιπτώσεις της.

- η κατηγορία **Cardinality**, που οι περιπτώσεις της είναι εκφράσεις που περιγράφουν τον πληθάριθμο του συνόλου των instances που επιτρέπεται να έχει μία κλάση ή μια κατηγορία σχέσεων.

- η κατηγορία **Behavior**, που οι περιπτώσεις της είναι συγκεκριμένα διαγράμματα μετάβασης καταστάσεων (βλ. σελ 62)

- η κατηγορία γνωρισμάτων **Attributes** χρησιμοποιείται για να περιγράψουμε τα γνωρίσματα μιας κλάσης, τις μεταβλητές που περιέχει.

- η κατηγορία γνωρισμάτων **Operations** χρησιμοποιείται για να περιγράψουμε τις μεθόδους μιας κλάσης.

- η κατηγορία γνωρισμάτων **ExportControl** χαρακτηρίζει σχέσεις ή κλάσεις και χρησιμοποιείται για να εκφράσει την πολιτική πρόσβασης που επιβάλλει μία κλάση στα γνωρίσματα και στις μεθόδους της. Περιπτώσεις αυτής της κατηγορίας είναι οι εξής:

  -- **public access**, που δηλώνει την έλλειψη περιορισμών πρόσβασης

  -- **protected access**, που δηλώνει ότι πρόσβαση έχουν η ίδια η κλάση, οι υποκλάσεις και οι friend κλάσεις. Στα διαγράμματα παρουσιάζεται σαν μία κάθετη γραμμή πριν από το όνομα του γνωρίσματος ή της μεθόδου.

  -- **private access**, που δηλώνει ότι δικαίωμα πρόσβασης έχουν μόνο η ίδια η κλάση και οι friend κλάσεις. Στα διαγράμματα παρουσιάζεται σαν δύο κάθετες γραμμές πριν από το όνομα του γνωρίσματος ή της μεθόδου.

  -- **implementation access** που δηλώνει ότι δικαίωμα πρόσβασης έχει μόνο η ίδια η κλάση. Στα διαγράμματα παρουσιάζεται σαν τρεις κάθετες γραμμές πριν από το όνομα του γνωρίσματος ή της μεθόδου.

- η κατηγορία containment που μπορεί να χαρακτηρίζει μια σχέση τύπου uses, και οι περιπτώσεις αυτής της κατηγορίας γνωρισμάτων είναι δύο : by reference, όταν το περιεχόμενο αντικείμενο είναι ουσιαστικά ένας δείκτης ή μια αναφορά και by value όταν το περιεχόμενο αντικείμενο περιέχεται σαν τιμή ή σαν τμήμα στο άλλο.

- η κατηγορία γνωρισμάτων **Class** αποδίδεται σε σχέσεις και δηλώνει ότι η συγκεκριμένη σχέση ανήκει σε μια κλάση σχέσεων.

- η κατηγορία γνωρισμάτων **Role** εισάγεται από το αντικείμενο RelationIcon και χρησιμοποιείται για να εκφράσει το ρόλο που παίζει κάθε μία από τις κλάσεις που σχετίζονται μέσω της σχέσης που έχει αυτό το γνώρισμα.

- η κατηγορία γνωρισμάτων **Key** χρησιμοποιείται για να αποδοθούν ιδιότητες κλειδιού (μοναδική τιμή του attribute για κάθε περίπτωση της κλάσης ή της σχέσης) σε κάποιο από τα attributes μιας κλάσης.

Οι ιδιότητες που αποδίδονται στις κλάσεις στη φάση του specification αλλά δεν αντιστοιχούν σε icons, δεν περιλαμβάνονται στα Objects που ανήκουν στην ιεραρχία του ClassDiagramElements.Τέτοιες ιδιότητες είναι : *Persistence, Concurrency, SpaceComplexity, Qualification, Preconditions,PostConditions,Exceptions*

Διαγράμματα αντικειμένων

Στη μεθοδολογία του Booch αποτελούν μέρος της λογικής σχεδίασης του συστήματος. Τα συστατικά στοιχεία αυτού του τύπου διαγράμματος είναι τα αντικείμενα (objects) και οι συσχετίσεις μεταξύ τους Στο δικό μας μοντέλο παριστάνουμε τα εικονίδια αντικειμένων με την κλάση *ObjectIcon*, και τις συσχετίσεις μεταξύ αντικειμένων με το Telos_Object *LinkIcon*.

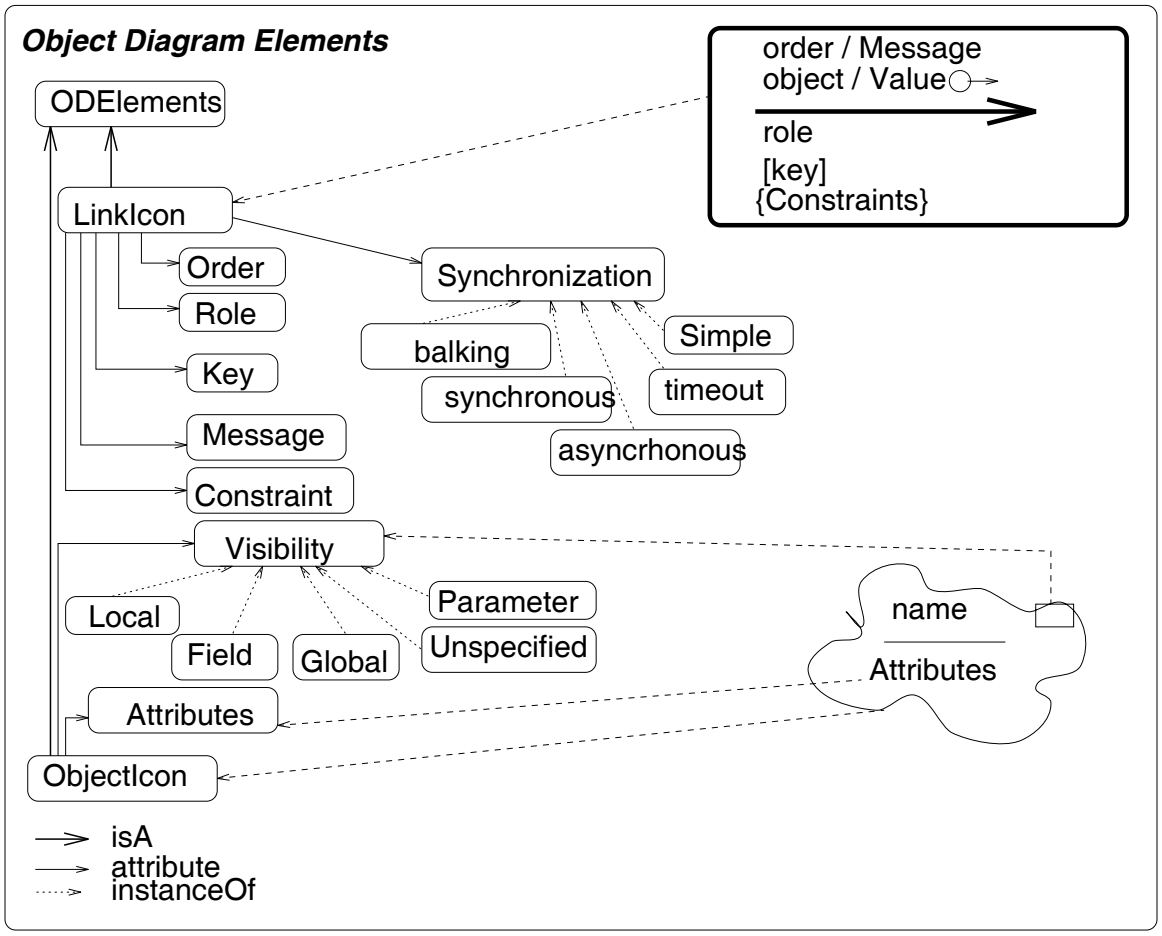
Ο στόχος κάθε διαγράμματος αντικειμένων είναι να παρακολουθηθεί (ιχνηλατηθεί) η εκτέλεση κάποιας λειτουργίας του συστήματος.

Στο σχήμα 6.10 βλέπουμε τα Telos_Objects που χρησιμοποιήσαμε για να περιγράψουμε διαγράμματα αντικειμένων και την αντιστοίχισή τους με τα εικονίδια του αντίστοιχου διαγράμματος.

- Στο διάγραμμα παριστάνονται για κάθε αντικείμενο το όνομα και τα attributes του, τα οποία έχουν τη μορφή **A:C** ή **A** ή **:C** όπου το A δηλώνει το όνομα και το C την κλάση στην οποία ανήκει το γνώρισμα. Το αντικείμενο *ObjectIcon* εισάγει μια κατηγορία γνωρισμάτων, τα *Attributes*,για να δηλωθούν τα γνωρίσματα του αντικειμένου. Στο διάγραμμα δεν είναι απαραίτητο να εμφανίζονται όλα τα γνωρίσματα του αντικειμένου - υπενθυμίζουμε εδώ ότι τα διαγράμματα γενικώς, παριστάνουν *όψεις* του συστήματος.

- Η *κατεύθυνση* , που στο διάγραμμα φαίνεται με ένα επιπλέον βέλος πάνω από τη γραμμή που συνδέει τα αντικείμενα δηλώνεται δίνοντας στο *LinkIcon* τα γνωρίσματα **orig** και **dest**.

- Τα αντικείμενα θεωρούμε ότι επικοινωνούν ανταλλάζοντας μηνύματα . Ο τύπος του συγχρονισμού δηλώνεται με το γνώρισμα **Synchronization** που



Σχήμα 6.10: Το μετα μοντέλο για τα Object Diagrams

παίρνει μία ή περισσότερες από ένα σύνολο με 5 διαφορετικές διακριτές τιμές.

-- **simple** για να δηλώσει απλό ακολουθιακό πέρασμα μηνυμάτων.

-- **balking** που δηλώνει ότι ο client θα σταματήσει να περιμένει αν ο server δεν μπορεί να ανταποκριθεί αμέσως

-- **asynchronous** δηλώνει ότι ο client στέλνει το μήνυμά του στον server και δεν περιμένει για την απάντηση, αλλά συνεχίζει τη δουλειά του.

-- **synchronous** δηλώνει ότι ο client περιμένει αντίδραση από τον server για να αντιδράσει

-- **timeout** δηλώνει ότι ο client τα σταματήσει να περιμένει την απάντηση του server όταν περάσει ένα συγκεκριμένο χρονικό διάστημα.

- Τα γνωρίσματα της κατηγορίας **Role** εκφράζουν την κατεύθυνση προς την οποία γίνεται η ροή των δεδομένων, ποιος παίξει το ρόλο του client και ποιός του server.

- **Visibility**(ορατότητα), καθορίζει τον τρόπο με τον οποίο ένα object βλέπει ένα άλλο. Οι περιπτώσεις και αυτής της κατηγορίας γνωρισμάτων ανήκουν σε συγκεκριμένο διακριτό σύνολο. Έτσι, η ορατότητα μπορεί να είναι:
  - -- **Local** που δηλώνει ότι το ένα αντικείμενο είναι τοπικό στην εμβέλεια του άλλου
  - -- **Field** που δηλώνει ότι το ένα αντικείμενο αποτελεί πεδίο του άλλου
  - -- **Global** που δηλώνει ότι το αντικείμενο είναι ορατό γιατί βρίσκεται μέσα σε κάποια συνολική εμβέλεια
  - -- **Unspecified**, για να περιγράψουμε περιπτώσεις όπου η ορατότητα μεταξύ των δύο αντικειμένων δεν έχει καθοριστεί.
  - -- **Parameter** που δηλώνει ότι το ένα αντικείμενο περνιέται σαν παράμετρος στο άλλο.

Διαγράμματα μετάβασης καταστάσεων

Τα διαγράμματα μετάβασης καταστάσεων περιγράφουν τη συμπεριφορά μιας κλάσης όσον αφορά τις μεταβάσεις καταστάσεών της και την αντίδρασή της σε events.

Ο συμβολισμός (notation) που χρησιμοποιεί ο Booch έχει υιοθετηθεί από τον Harel [HK92], είναι δηλαδή τα διαγράμματα κάτι ανάλογο με τα Statecharts που χρησιμοποιούνται στο STATEMATE [Har90].

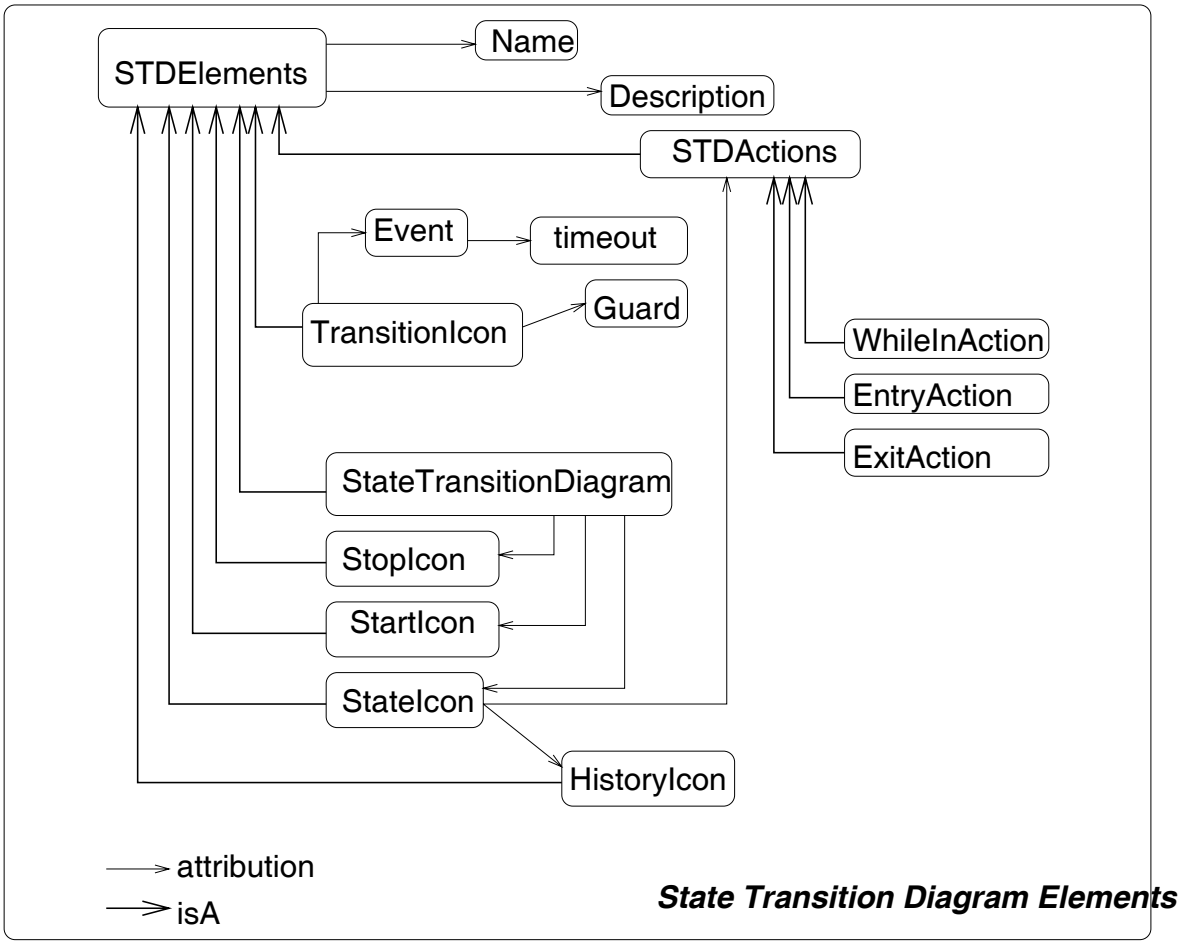
Στα διαγράμματα αυτά βασικές οντότητες είναι οι καταστάσεις και οι μεταβάσεις μεταξύ καταστάσεων. Οι διάφοροι τύποι τους και τα γνωρίσματά τους όπως αποδόθηκαν στο μοντέλο μας περιγράφονται στη σελίδα 93 (στο παράρτημα με την περιγραφή του μοντέλου σε Telos) ενώ στο σχήμα 6.11 βλέπουμε σχηματικά τις παραπάνω οντότητες καθώς και τις σχέσεις γενίκευσης και απόδοσης γνωρισμάτων που ισχύουν μεταξύ τους.

Ενα εικονίδιο κατάστασης *StateIcon* μπορεί να σχετίζεται με κάποιες ενέργειες *Actions* που συμβαίνουν μόλις εισέρχεται εξέρχεται ή όσο παραμένει το αντικείμενο στην κατάσταση αυτή. Οι ενέργειες θεωρούμε ότι είναι Operations που εκτελούνται σε μηδενικό χρόνο. Actions είναι κλήσεις μεθόδων, πυροδοτήσεις άλλων γεγονότων ή ξεκίνημα-σταμάτημα κάποιας δραστηριότητας (*Activity*)

Μία μετάβαση χαρακτηρίζεται από ένα Event μόνο του ή μαζί με μια συνθήκη (*STDGuard* και μπορεί να έχει (όχι απαραίτητα) timeout.

Διαγράμματα αλληλεπίδρασης

Χρησιμεύουν για να παρακολουθήσουμε ένα σενάριο, όπως και τα διαγράμματα αντικειμένων. Δείχνουν όμως καλύτερα τη σχετική σειρά των μηνυμάτων που περνάνε, με το επιπλέον χαρακτηριστικό του FocusOfControl.



Σχήμα 6.11: Το μετα μοντέλο για τα State TransitionDiagrams

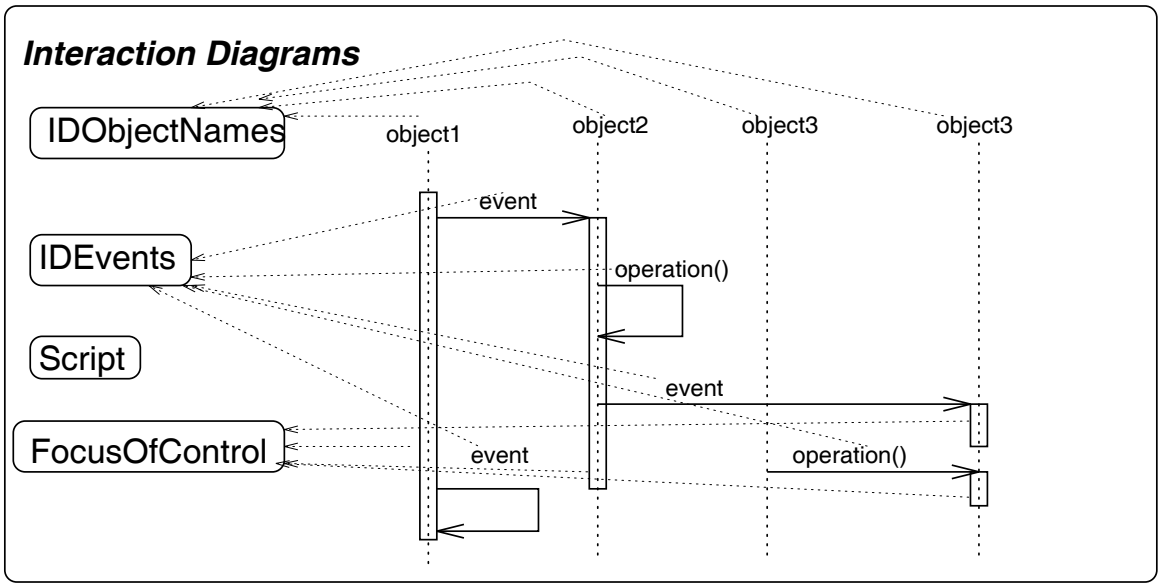
Είναι πιο εύκολα να χρησιμοποιούνται στην αρχή της σχεδίασης γιατί εστιάζουν την προσοχή στα αντικείμενα όταν ακόμη δεν έχουν καθοριστεί οι μέθοδοι και τα πρωτόκολλα των κλάσεων. Τα διαγράμματα αλληλεπίδρασης παριστάνονται σε μορφή πίνακα. Οι στήλες αντιστοιχούν σε αντικείμενα και οι γραμμές παριστάνουν events με τη σειρά που εμφανίζονται. Στο σχήμα 6.12 βλέπουμε ότι το διάγραμμα αλληλεπίδρασης εισάγει σε σχέση με το ObjectDiagram την έννοια του script, που είναι κείμενο του σεναρίου,και του FocusOfControl, που δείχνει σχηματικά πόσο χρόνο βρίσκεται ο έλεγχος σε κάθε αντικείμενο.

Στο μοντέλο των διαγραμμάτων μας λοιπόν, τα *InteractionDiagrams* είναι υποκλάση του *ObjectDiagram* με δύο επιπλέον *Elements*

Διαγράμματα Modules

Μέρος της φυσικής άποψης του συστήματος (physical view) Περιγράφουν την κατανομή των κλάσεων και των αντικειμένων σε modules. Αντιστοιχούν στην κατανομή της υλοποίησης του συστήματος σε αρχεία και βιβλιοθήκες.

Διαγράμματα Επεξεργασίας



Σχήμα 6.12: Διαγράμματα αλληλεπίδρασης

Μέρος της φυσικής άποψης του συστήματος (physical view), επίσης. Παριστάνουν την κατανομή των διεργασιών στους επεξεργαστές του συστήματος και τις συνδέσεις μεταξύ processors και devices. Συνήθως είναι πολύ στενά συνδεμένα με τους τεχνολογικούς περιορισμούς του συστήματος και ο μηχανικός λογισμικού δεν έχει μεγάλη δυνατότητα επιλογών σ'αυτό το σημείο και άρα ούτε και ανάγκη καθοδήγησης. Γι'αυτό δεν τα αναλύουμε περεταίρω στο μοντέλο του προϊόντος μας.

6.2.4 Παρατηρήσεις

Η διαδικασία που μελετάμε είναι μια διαδικασία ανάπτυξης λογισμικού.Την παρακολουθούμε από τα στάδια της ανάλυσης μέχρι τη σχεδίαση και υλοποίηση.

Το προϊόν που μετασχηματίζεται κατά τη διαδικασία αυτή είναι οι απαιτήσεις που γίνονται διαγράμματα - κλάσεις μοντέλα και τελικά πηγαίος κώδικας C++.

Έτσι, έχουμε βασικά δύο όψεις του προϊόντος: τη σχεδιαστική και την υλοποίησης. Όσον αφορά τη φάση της υλοποίησης, το μοντέλο του προϊόντος κατασκευάζεται με αναδιάρθρωση από τελικό κώδικα και παριστάνεται σύμφωνα με ένα μοντέλο για τη C++ της Telos, βλ. StaticAnalyzer [CJMV95]. Όσον αφορά τη φάση του σχεδιασμού το μοντέλο που περιγράφει το προϊόν, βασίζεται στο μοντέλο διαγραμμάτων που κατασκευάζουμε με βάση το συμβολισμό που χρησιμοποιεί ο Booch για τα διάφορα διαγράμματα που περιγράφουν το σύστημα.

Για να έχουμε μια ενοποιημένη άποψη για το προϊόν, καλό είναι να συσχετί-

ζουμε στο μοντέλο μας το προϊόν μεταξύ των δύο αυτών φάσεων. Η συσχέτιση αυτή μπορεί να γίνει άμεσα αποκαθιστώντας μια σχέση μεταξύ των οντοτήτων των διαγραμμάτων και των οντοτήτων του C++Model του StaticAnalyzer Συγκεκριμένα στις οντότητες εκείνες του μοντέλου προϊόντος που υλοποιούνται με συγκεκριμένες δομές της C++ αποδίδουμε ένα γνώρισμα implementedBy, με προορισμό την αντίστοιχη οντότητα του μοντέλου της C++. Σύμφωνα με το μοντέλο της Telos αυτό γίνεται χωρίς πρόβλημα. Υπάρχουν όμως περιπτώσεις που η συγκεκριμένη υλοποίηση μας περιορίζει, απαγορεύοντας τον προορισμό ενός γνωρίσματος να είναι γνώρισμα. για παράδειγμα , οι σχέσεις π.χ. κληρονόμησης μεταξύ κλάσεων σ'ένα διάγραμμα στο μοντέλο της C++ παριστάνονται σαν γνώρισμα της μιας από τις σχετιζόμενες κλάσεις.Σ'αυτές τις περιπτώσεις δεν μπορούμε να ορίσουμε το γνώρισμα implementedBy για τα RelationIcons. Μπορούμε όμως να ορίσουμε το αντίστροφο , δηλαδή στα γνώρισματα οντοτήτων του μοντέλου της C++ που υλοποιούν σχέσεις που παριστάνονται σε διαγράμματα, αποδίδουμε ένα γνώρισμα implements με προορισμό το αντίστοιχο ClassIcon. Δεν καταργούσε όμως το implementedBy γιατί οι περιγραφές των υλοποιήσεων σε C++ παράγονται αυτόματα , και είναι ευκολότερο να επεμβαίνουμε στις περιγραφές των διαγραμμάτων για να ορίσουμε τις περιπτώσεις του γνωρίσματος.

6.3 Καθοδήγηση μέσω ομοιότητας κατά την εκτέλεση του μοντέλου

Για το συγκεκριμένο μεταμοντέλο διαδικασίας έχει γίνει δουλειά ([SSRG96]) όσον αφορά την αυτοματοποίηση του "αλγορίθμου" διάσχισης του δέντρου της διαδικασίας, όπου ο μηχανισμός καθοδήγησης και το μοντέλο διαδικασίών στηρίζεται στο σύστημα διαχείρισης βάσης Δεδομένων O_2. Στη δική μας περίπτωση το μοντέλο είναι αποθηκευμένο σε Telos και δεν επιχειρούμε πλήρη αυτοματοποίηση της εκτέλεσης. Όπως τονίστηκε και σε προηγούμενα κεφάλαια, εμείς εστιάζουμε στην καθοδήγηση σε περιπτώσεις μεθοδολογικών διλημμάτων, και όχι στην αυτοματοποίηση της διαδικασίας.

6.3.1 Προσομοίωση της "εκτέλεσης" του μοντέλου

Έχοντας περιγράψει το μοντέλο της διαδικασίας σαν περίπτωση του μεταμοντέλου, η εκτέλεση του και κατά συνέπεια και η παραγωγή ιχνών προϋποθέτει τη διάσχιση του δέντρου της διαδικασίας με τον αλγόριθμο που περιγράφτηκε στη σελ 44. Όταν εκτελούμε το μοντέλο κινούμαστε σε ένα επίπεδο αφαίρεσης χαμηλότερο από το μοντέλο. Κάθε βήμα-απόφαση στην πορεία μας περιγράφεται σαν περίπτωση κάποιου περιβάλλοντος απόφασης του μοντέλου διαδικασίας. Γι'αυτό λέμε ότι κατά την εκτέλεση του μοντέλου παράγουμε ίχνη (περιπτώσεις εκτέλεσης).

Η διάσχιση του δέντρου, γίνεται ως εξής : Χρησιμοποιούμε τον αλγόριθμο για να βρούμε ποιά είναι τα δυνατά επόμενα περιβάλλοντα απόφασης και αφού επιλέξουμε ένα περιβάλλον απόφασης φτιάχνουμε το ίχνος του σαν περίπτωση της κατάστασής του περιβάλλοντος αυτού. Για να καθορίζουμε λοιπόν το πως ακολουθείται ο αλγόριθμος αρκεί να περιγράψουμε πως παράγονται τα instances των διαφόρων περιβαλλόντων απόφασης.

Ο αλγόριθμος λέει ότι σε κάθε *περιβάλλον επιλογής* , για να προχωρήσουμε επιλέγουμε μία μόνο από τις εναλλακτικές λύσεις. Αυτό στο μοντέλο μας εισάγει τον περιορισμό ότι μία περίπτωση οποιουδήποτε *ChoiceContext* πρέπει να έχει ένα ακριβώς γνώρισμα της μετα-κατηγορίας *ChoiceContext.alternatives* Επειδή η ιδιότητα αυτή χαρακτηρίζει την μετακατηγορία *ChoiceContext.alternatives*, η απόδοση αυτής της ιδιότητας επιλέχτηκε να γίνει κάνοντας τη συγκεκριμένη μετα κατηγορία περίπτωση μιας γενικότερης κατηγορίας γνωρισμάτων που ορίστηκε για το σκοπό αυτό. Έτσι, ορίσαμε μια κατηγορία γνωρισμάτων σε επίπεδο M2_Class με όνομα *single_at_Token* και με αφετηρία και προορισμό το **Individual**, και στη συνέχεια ορίστηκε η *ChoiceContext.alternatives* να είναι

περίπτωση αυτής της μεταμετα κατηγορίας. Για τα ίχνη των περιβαλλόντων σχεδίου δεν χρειάστηκε να οριστεί κάτι ανάλογο, αφού τα γνωρίσματα στην Telos έχουν πολλαπλές περιπτώσεις by default.

Παρακάτω θα δούμε μερικές από τις δυνατότητες που μπορούμε να παρέχουμε στο χρήστη του συστήματός μας, όσον αφορά την καθοδήγησή του με βάση οποιαδήποτε διαδικασία που έχει οριστεί σαν περίπτωση του μεταμοντέλου διαδικασιών μας και έχει αποθηκευτεί στη βάση γνώσης Telos.

Παρουσίαση του δέντρου επιλογών

α. σε επίπεδο μοντέλου

Η παρουσίαση στον χρήστη μιας συνολικής εικόνας του δέντρου επιλογών ή οποιουδήποτε υποδέντρου του, δίνει στο χρήστη την πληροφορία του "τί μπορεί να γίνει από δω και πέρα" σύμφωνα με τη μέθοδο που ακολουθείται. Γι'αυτό έχει προστεθεί στην επαφή χρήσης του συστήματος σημασιολογικού ευρετηριασμού μια επιλογή **Process Guidance**, που δίνει το υποδέντρο της ιεραρχίας των περιβαλλόντων απόφασης, όπως έχουν οριστεί στο επίπεδο του μοντέλου της διαδικασίας, με αρχή το περιβάλλον απόφασης που δίνεται στο Query Target. Στην περίπτωση που στο πεδίο Query Target δίνεται η οντότητα Context του μεταμοντέλου, τότε παρουσιάζουμε όλο το δέντρο της διαδικασίας με μορφή ιεραρχίας από Contexts.

β. σε επίπεδο ιχνών Στην περίπτωση που στο πεδίο Query Target δίνεται ένα ίχνος, μπορούμε να δούμε όλο το δέντρο των αποφάσεων που πάρθηκαν ξεκινώντας από το συγκεκριμένο ίχνος, με την επιλογή *Trace Path* του μενού *ProcessMenu*

Πως γίνεται κάτι

Με δεδομένο ένα ίχνος της διαδικασίας η πληροφορία του πως οδηγηθήκαμε εκεί, δηλαδή το μονοπάτι προς τα πίσω δίνεται με την επιλογή *Trace Back* του μενού *ProcessMenu*

Εύρεση δυνατών επόμενων περιβαλλόντων απόφασης

Στην ερώτηση "τί μπορεί να είναι το επόμενο βήμα" η απάντηση δίνεται με μια πιο περιορισμένη άποψη του δέντρου της διαδικασίας που επιπλέον λαμβάνει υπόψιν της τον αλγόριθμο διάσχισης του δέντρου. Συγκεκριμένα, για να το υποστηρίξουμε αυτό έχουμε προσθέσει τις εξής προκαθορισμένες ερωτήσεις στην επαφή χρήσης του συστήματός μας.

- Όταν βρισκόμαστε σε περιβάλλον επιλογής, τότε τα δυνατά επόμενα βήματα είναι οι επιλογές που ορίζονται από το μοντέλο αφού λάβουμε υπόψιν μας τους περιορισμούς κατάστασης. Η ερώτηση αυτή υλοποιείται από το μενού *ProcessMenu* και συγκεκριμένα την επιλογή *ChoiceNext*

- Όταν βρισκόμαστε σε περιβάλλον σχεδίου, τότε τα δυνατά επόμενα βή-

ματα περιλαμβάνουν εκτός από τα ”παιδιά” του και τα (επόμενα των) αδέλφια του αν έχουν εκτελεστεί μια φορά τουλάχιστον τα παιδιά του. Η ερώτηση αυτή υλοποιείται από το μενού *ProcessMenu* και συγκεκριμένα α. την επιλογή *PlanNext* η οποία χρησιμεύει όσο θέλει ο χρήστης να επαναλαμβάνει τα βήματα του σχεδίου, και β. την επιλογή *Done with plan* που εκφράζει την πρόθεση να προχωρήσουμε με άλλο περιβάλλον απόφασης.

- Όταν βρισκόμαστε σε περιβάλλον εκτέλεσης, το επόμενο βήμα είναι ο πρώτος πρόγονος του που είναι περιβάλλον σχεδίου. Η ερώτηση αυτή υλοποιείται από το μενού *ProcessMenu* και συγκεκριμένα την επιλογή *ExecNext*

6.3.2 Υποστήριξη χρήσης αναλογικού συλλογισμού

Στις περιπτώσεις που έχουμε δυνατότητα επιλογής μπορεί να εφαρμοστεί ο μηχανισμός ανάκλησης παραδειγμάτων που περιγράφτηκε σε προηγούμενο κεφάλαιο.

Θεωρούμε ότι η βάση του αναλογικού συλλογισμού είναι η κατάσταση στην οποία βρίσκεται το προϊόν της διαδικασίας σε συνδυασμό με την πρόθεση που έχουμε για τη συνέχεια. Το πρόβλημα που παρουσιάζεται σ’αυτό το σημείο είναι ο τρόπος προσδιορισμού της τρέχουσας κατάστασης. Ξεκινώντας όμως τη διαδικασία από τη αρχή και ακολουθώντας τα βήματα που αυτή καθορίζει, μπορούμε να ξέρουμε ποια είναι η τρέχουσα κατάσταση. Με δεδομένη την περιγραφή της τρέχουσας κατάστασης και θεωρώντας ότι στην αποθήκη μας υπάρχουν ίχνη δηλαδή άλλες καταστάσεις που αποτελούν περιπτώσεις της ίδιας κατάστασης (από εκτέλεση του ίδιου ΜΔ) τότε, κατ’αρχήν παράγουμε, μέσω της ερώτησης MostSimilarInstance του similarity analyzer, μια ταξινομημένη κατ’αύξουσα ομοιότητα λίστα από όλες τις κατάλληλες αποθηκευμένες. Αυτή είναι μια λίστα με τα ανάλογα παραδείγματα. Τα οποία περιγράφουν βήματα της ίδιας διαδικασίας, από την τρέχουσα ή από προηγούμενες εφαρμογές. Στη συνέχεια ξεκινώντας από την ”καλύτερη περίπτωση”, συγκρίνω ανά δύο την τρέχουσα κατάσταση με κάθε μια από τις αποθηκευμένες Έτσι εντοπίζω τις συγκεκριμένες αναλογίες, που με καθοδηγούν στην επιλογή της πιο κοντινής. Την απόφαση που αντιστοιχεί στην πιο κοντινή μπορώ να τη διαλέξω σαν καλύτερη για το τωρινό δίλημμα.

Πολύ συχνά εμφανίζεται και το εξής πρόβλημα στους μηχανικούς λογισμικού που ακολουθούν μια συγκεκριμένη μέθοδο παραγωγής λογισμικού: Σύμφωνα με τη μέθοδο, ένα διάγραμμα μετατρέπεται κατά την υλοποίηση σε περιγραφή σε κάποια συγκεκριμένη γλώσσα προγραμματισμού. Το διάγραμμα είναι απο-

τέλεσμα της ανάλυσης, και δεν έχει λάβει υπόψιν του σχεδιαστική πληροφορία. Η υλοποίση λοιπόν μπορεί να ποικίλει και συνεπώς, είναι χρήσιμο να μπορεί να δει κανείς εναλλακτικές υλοποιήσεις παρομοίων μοντέλων συστημάτων. Τότε μπορούμε να παρέχουμε βοήθεια δείχνοντάς του παρόμοια διαγράμματα πώς υλοποιήθηκαν τελικά (χρησιμοποιώντας τη σύνδεση με τα γνωρίσματα implements και implementedBy που ορίσθηκαν στην προηγούμενη παράγραφο)

6.3.3 Παραγωγή ιχνών

Η έννοια του ίχνους στο μοντέλο

Τα ίχνη της διαδικασίας είναι περιπτώσεις των περιβαλλόντων απόφασης του μοντέλου που την περιγράφει.[sygkekrimeno paradeigma kai sxhma???] Πιο συγκεκριμένα, ένα *ίχνος* (*Trace*) στο μετα μοντέλο μας είναι περίπτωση της οντότητας *Περιβάλλον απόφασης* (Context). Περιπτώσεις της οντότητας *Trace* είναι οι περιπτώσεις των περιπτώσεων τον οντοτήτων PlanContext, ChoiceContext και ExecutionContext. Η οντότητα *Trace* βρίσκεται στο ίδιο επίπεδο αφαίρεσης με τις οντότητες που περιγράφουν τα περιβάλλοντα απόφασης ενός μοντέλου διαδικασίας(όπως τα περιβάλλοντα απόφασης για τη διαδικασία BOOD).

Επιπλέον, κάθε περίπτωση των (M1) κλάσεων *PlanContext, ChoiceContext* και*ExecutionContext* δηλαδή όλες οι κλάσεις που παριστάνουν περιβάλλοντα απόφασης στο μοντέλο της διαδικασίας ανάλυσης του Booch. είναι υποκλάση (υποσύνολο) της κλάσης *Trace.*

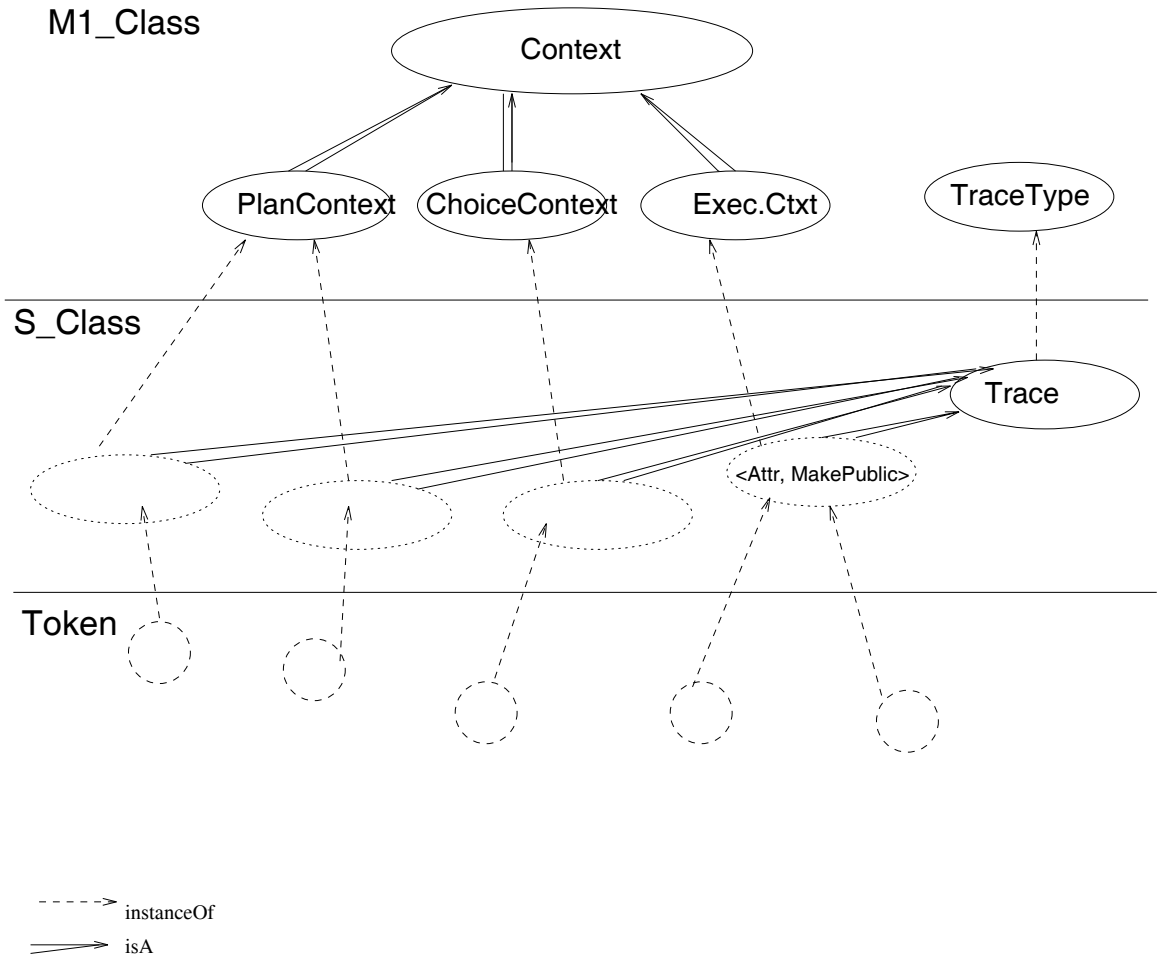
Επομένως οι περιπτώσεις της κλάσης *Trace*, δηλαδή τα ίχνη είναι αντικείμενα σε επίπεδο Token, που εκτός από την κλάση του συγκεκριμένου Context που ανήκουν, ανήκουν και στην *Trace* αφού η σχέση isA έχει σημασία υποσυνόλου)

Σε επίπεδο M1_Class ορίσαμε την κλάση TraceType που έχει σαν περιπτώσεις της όλες τις περιπτώσεις των PlanContext, ChoiceContext και ExecutionContext. Η TraceType είναι η συζυγής μετακλάση (associated Superclass) της *Trace* και εννοιολογικά είναι ένα υποσύνολο του δυναμοσυνόλου του συνόλου των περιπτώσεων της *Trace.*

Σχηματικά η συσχετίσεις των εννοιών που προαναφέρθηκαν, φαίνονται στο σχ. 6.13.

Πώς Δημιουργούμε ίχνη

Όπως ειπώθηκε προηγουμένως, τα ίχνη είναι περιπτώσεις των Contexts. Πιο συγκεκριμένα :



Σχήμα 6.13: Μοντελοποίηση ιχνών(*Traces*)

- ίχνη των PlanContexts έχουν μία ή πολλαπλές τιμές στα γνωρίσματα της μετα-κατηγορίας *PlanContext.depgraph* Κάθε μία παριστάνει μια διαφορετική εφαρμογή του συγκεκριμένου περιβάλλοντος απόφασης που εκτελέστηκε στα πλαίσια της τρέχουσας διαδικασίας.

- ίχνη των ChoiceContexts έχουν μία μόνο τιμή στο γνώρισμα της μετα-κατηγορίας *ChoiceContext.alternatives* που παριστάνει ποιά από τις εναλλακτικές λύσεις ακολουθήθηκε.

- ίχνη των ExecutionContexts έχουν περιπτώσεις της κατηγορίας γνωρισμάτων changes που παίρνουν τιμές σε περιπτώσεις της κλάσης Product. Αυτά καθορίζουν και το ποιο είναι το επόμενο περιβάλλον απόφασης αφού καθορίζουν και ποια είναι η επόμενη δυνατή κατάσταση.

Προς το παρόν η διαδικασία εισαγωγής κατάλληλων ιχνών στη βάση δεν είναι αυτοματοποιημένη. Μπορεί όμως να γίνει, με τη σύνδεση της παρακολούθησης

της εκτέλεσης με ένα εργαλείο παραγωγής ιχνών, ή ακόμη και με διαλογική εισαγωγή της τρέχουσας κατάστασης στη βάση σαν περίπτωση της κατάστασης του τρέχοντος περιβάλλοντος απόφασης.(Η εισαγωγή σύνθετων αντικειμένων διευκολύνεται με τα δελτία εισαγωγής δεδομένων [Δασ96]).

Η ύπαρξη μιας ικανής αποθήκης ιχνών είναι σημαντική για την επιτυχία της καθοδήγησής μας. Γι'αυτό επιχειρούμε να ανακατασκευάσουμε τη διαδικασία παραγωγής κάποιου προϊόντος λογισμικού έχοντας σαν είσοδο το τελικό προϊόν. Έτσι θα έχουμε αυτόματη κατασκευή ιχνών χωρίς στην πραγματικότητα να ακολουθήσουμε τη διαδικασία, δηλαδή σε περιπτώσεις που η πορεία μας δεν έχει καταγραφεί. Αυτή η λύση δίνει ικανοποιητικά αποτελέσματα παραβλέποντας το ότι προστίθενται νέα επιχειρήματα κατά την εκτέλεση της διαδικασίας. Αυτό δε σημαίνει ότι τα αγνοούμε, απλά μας δίνει ένα τρόπο να αντιμετωπίσουμε το γεγονός ότι πολύ συχνά για κάποιο λόγο δεν καταγράφονται. Επεξεργαζόμαστε τα αρχεία που περιγράφουν την υλοποίηση του συστήματος (το τελικό προϊόν) και παίρνουμε τη στατική ανάλυση. Στη συνέχεια εντοπίζουμε ποια περιβάλλοντα απόφασης έχουν ήδη εκτελεστεί ως εξής: Εκεί που το μοντέλο διαδικασίας δίνει επιχειρήματα επιλογής μεταξύ εναλλακτικών λύσεων, μπορούμε να θεωρήσουμε ότι αυτά ίσχυσαν αφού εντοπίσουμε ότι επιλέχτηκε η συγκεκριμένη απόφαση. Για παράδειγμα, το επιχείρημα που γίνει η μέθοδος για να γίνει ένα γνώρισμα Public είναι ότι θέλουμε να έχουν πρόσβαση σ'αυτό όλοι οι clients της κλάσης. Εντοπίζοντας εμείς ένα γνώρισμα Public μπορούμε να συμπεράνουμε ότι η κατασκευή του πέρασε από το περιβάλλον απόφασης < Attribute, Confirm> της διαδικασίας και να παράγουμε το αντίστοιχο ίχνος, του οποίου η περίσταση εξαρτάται από το συγκεκριμένο γνώρισμα και απόφαση Confirm, με το επιχείρημα που παρέχει η μέθοδος γι'αυτή την απόφαση ("All clients should have Acces to this Attribute").

Βέβαια, αυτό σημαίνει τη δημιουργία και άλλων ιχνών, που οδηγούν σ'αυτό άλλα προς το παρόν επειδή δεν εξετάζουμε μονοπάτια από ίχνη αλλά τα ίχνη απομονωμένα, αυτό μας αρκεί.

Κεφάλαιο 7

Επίλογος

7.1 Εφαρμοσιμότητα

Στην εργασία αυτή παρουσιάστηκε μια πρόταση για καθοδήγηση διαδικασιών παραγωγής περιγραφών που χρησιμοποιεί την αναλογία μεταξύ καταστάσεων και τελικά προϊόντων.

Σημαντικό χαρακτηριστικό όμως των διαδικασιών αυτών είναι ο μη τυποποιημένος χαρακτήρας τους καθώς και το γεγονός ότι τα προϊόντα τους (οι περιγραφές) είναι συνήθως διαγράμματα ή κείμενο σε φυσική γλώσσα, κυρίως στα αρχικά στάδια.

Ακόμη και αν πρόκειται για ημιτυπικές περιγραφές, το ουσιαστικό μας πρόβλημα εντοπίζεται στο χειρισμό των καταγεγραμμένων επιχειρημάτων που ούτως η άλλως δεν παρουσιάζουν δομημένη μορφή μια και είναι προτάσεις σε φυσική γλώσσα. Παρόλαυτα, όσον αφορά τα επιχειρήματα, αφενός τα διακρίνουμε καταρχήν σαν υποστηρικτικά ή αποτρεπτικά για κάποια απόφαση και αφετέρου μπορούμε να καταγράψουμε τις συσχετίσεις που έχουν -αν αυτές υπάρχουν - με τα προϊόντα τα οποία επηρεάζονται υπό τις εκάστοτε αποφάσεις. Έτσι, όσον αφορά το μοντέλο αναλογικής ομοιότητας, τα επιχειρήματα παίζουν ένα διπλό ρόλο: Αφενός υποστηρίζουν ή αντιτίθενται σε κάποια απόφαση και αφετέρου συσχετίζονται με στοιχεία της περιγραφής πάνω στα οποία μπορούμε να εντοπίσουμε αναλογίες. Αφού το προϊόν μας είναι η περιγραφή, και κάνουμε την παραδοχή ότι υπάρχει σε τυπική μορφή, μπορούμε να υπολογίσουμε ομοιότητα. Η παραδοχή αυτή είναι αρκετά λογική γιατί εφόσον ο κύριος στόχος των διαδικασιών που εξετάζουμε είναι η παραγωγή περιγραφών- μοντέλων μπορούμε να θεωρήσουμε ότι η πρόταση καθοδήγησής μας εφαρμόζεται σε κάποιο στάδιο της διαδικασίας όπου υπάρχουν κάποιες τυπικές περιγραφές του προϊόντος. Ένας λόγος που υποστηρίζει αυτή τη θεώρηση είναι το ότι οι διαδικασίες παραγωγής λογισμικού γενικότερα προχωρούν σπειροειδώς πρόκει-

ται για επαναληπτικές και αυξητικά εξελισσόμενες διαδικασίες (iterative and incremental) και όχι για γραμμικές, αφού η συνεχής αλλαγή και βελτίωση είναι ενδογενές χαρακτηριστικό της διαδικασίας παραγωγής λογισμικού. Επιπλέον, το συγκεκριμένο παράδειγμα της μεθόδου του Booch που αναπτύξαμε έδειξε ότι δεν είναι απαραίτητο να είμαστε στο τέλος της περιγραφής των απαιτήσεων αφού έχουμε παραστήσει τυπικά και για τα ενδιάμεσα διαγράμματα.

Συμπερασματικά λοιπόν μπορούμε να πούμε ότι η μέθοδός μας είναι εφαρμόσιμη σε μεγάλο μέρος του κύκλου ανάπτυξης ενός συστήματος, και συγκεκριμένα στο μεγαλύτερο μέρος της φάσης της επεξεργασίας των απαιτήσεων.

7.2 Αποτίμηση

Θα εκτιμήσουμε τώρα την εφαρμογή της μεθόδου μας. Σε σύγκριση με την καθοδήγηση που παρέχουν άλλα περιβάλλοντα καθοδήγησης, εμείς δεν προτείνουμε τρόπο αυτόματης ερμηνείας του μοντέλου. Το δικό μας process definition document δεν ερμηνεύεται αυτόματα και δυναμικά από κάποιο μηχανικό διερμηνέα, ούτε όμως είναι αδρανές όπως μια περιγραφή σε χαρτί. Παρόλο που για την αναπαράστασή του χρησιμοποιούμε μια γλώσσα παράστασης στατικής πληροφορίας, παρέχουμε στοιχειώδεις λειτουργίες καθοδήγησης.

Ο χρήστης κινείται σύμφωνα με τη διαδικασία ακολουθώντας το γράφο που την περιγράφει και παράλληλα δημιουργεί την συγκεκριμένη υλοποίηση της διαδικασίας. Όσο δεν έχει ανάγκη να δει παραδείγματα βέβαια, χρησιμοποιεί - διαβάζει το μοντέλο από τη βάση της Telos όπως θα το διάβαζε και από κάποιο κείμενο, ίσως με καλύτερα οργανωμένο τρόπο. Όταν όμως βρεθεί σε κάποιο σημείο επιλογής, τότε η βοήθεια που του παρέχουμε είναι ενεργή. Μπορεί να ζητήσει να δει ανάλογα παραδείγματα που θα του διαλευκάνουν τις εναλλακτικές λύσεις και θα του δώσουν κάποια πρόταση για τη συνέχεια.

Αξίζει να τονίσουμε εδώ ότι η χρήση των περιπτώσεων εφαρμογής μιας διαδικασίας, είναι σημαντική για την διατήρηση της συνέχειας σε ομάδες δουλειάς που αναλαμβάνουν την κατασκευή πολλών έργων λογισμικού. Δίνεται έτσι η δυνατότητα στην ομάδα να διατηρεί τη μνήμη από προηγούμενες εφαρμογές, και παρέχονται στους νέους σχεδιαστές παραδείγματα από διαφορετικές εφαρμογές, πράγμα που διευκολύνει την ενσωμάτωσή τους στο πνεύμα της ομάδας.

7.3 Αδυναμίες- επεκτάσεις- μελλοντικές κατευθύνσεις

Ένα λεπτό σημείο στην όλη προσέγγιση , είναι η προέλευση των παραδειγμάτων. Θεωρούμε ότι μπορεί να κατασκευαστεί απο την αρχή μια βάση με χαρακτηριστικά παραδείγματα για τις περιστάσεις διλημμάτων που εμφανίζονται στη διαδιικασία, πράγμα που εμείς επιχειρήσαμε με τη μέθοδο του Booch. Φυσικά ,για τον εμπλουτισμό της αποθήκης παραδειγμάτων, χρειάζεται επιλεκτική ιχνοληψία της εφαρμογής της διαδικασίας, αλλά αυτό δε μας απασχόλησε στα πλαίσια αυτής της εργασίας.

Εμείς εστιάσαμε στο κομμάτι εκείνο της καθοδήγησης που αφορά την ανάκληση σχετικών παραδειγμάτων. Για να αποτελέσει όμως αυτό το κομμάτι ένα τμήμα λειτουργικά ενταγμένο σε ένα σύστημα υποστήριξης της διαδικασίας ανάπτυξης λογισμικού χρειάζεται να υπάρχουν κάποιες προϋποθέσεις. Για να μπορεί να αυτοματοποιηθεί η εκτέλεση και η ιχνοληψία απαιτούνται δυνατότητες ενεργούς βάσης δεδομένων που να υποστηρίζει εκδόσεις.

Όσον αφορά την εκτέλεση, πιθανές επεκτάσεις της μεθόδου μας περιλαμβάνουν σύνδεση με γραφικά εργαλεία (που υπάρχουν στην αγορά) για την υποστήριξη της κατασκευής διαγραμμάτων. Θα μπορούσαμε να περιλάβουμε σ'αυτή την κατεύθυνση των επεκτάσεων και την ανάπτυξη γραφικών εργαλείων που εκτός από υποστήριξη της εκτέλεσης έχουν και δυνατότητα ιχνοληψίας, και την κατασκευή μεταφραστών που μετατρέπουν τα ίχνη των εργαλείων σε περιγραφές διαγραμμάτων όπως τις έχουμε περιγράψει στο κεφάλαιο 6. (Τα είδη διαγραμμάτων που χρησιμοποιεί ο Booch χρησιμοποιούνται και σε άλλες μεθόδους ανάλυσης και σχεδίασης.)

Όσον αφορά τώρα την ιχνοληψία, για να είναι αποδοτική πρέπει να υποστηρίζεται από τη δυνατότητα αποθήκευσης στην ίδια βάση διαφορετικών (διαδοχικών) εκδόσεων του προϊόντος. Επιπλέον μπορούμε να εμπλουτίσουμε την αποθήκη των ιχνών μας με περιπτώσεις όπου από το τελικό προϊόν μπορούμε να συμπεράνουμε τις αποφάσεις που πάρθηκαν κατά την σχεδίασή του. Χωρίς αυτές τις επεκτάσεις, μπορούμε να κατασκευάσουμε ίχνη από εκτελέσεις ενός μοντέλου διαδικασίας εισάγοντας με διαλογικό τρόπο (χρησιμοποιώντας τα δελτία εισαγωγής δεδομένων) όπου θα δημιουργούμε τις καταστάσεις σαν σύνθετα αντικείμενα ή γράφοντας σε Telos τις περιγραφές τους.

Ένα ανοιχτό θέμα [DBC88, SJ94] που άπτεται της εργασίας αυτής, είναι η χρήση της αναλογικής ομοιότητας όχι μόνο μεταξύ ιχνών, αλλά και μεταξύ μοντέλων διαδικασιών.Κάτι τέτοιο θα χρησίμευε στην εφαρμογή πολλαπλών μεθόδων στο ίδιο έργο, εντοπίζοντας τα σημεία στα οποία οι μεθοδολογίες

μπορούν να συνεργάζονται.

Βιβλιογραφία

[Ντα93] Κώστας Νταντουρής. Βιβλιοθήκη στοιχειωδών ερωτηματικών συναρτήσεων και επεξεργασία ερωτήσεων στη γλώσσα Telos. Master's thesis, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Ιούλιος 1993.

[Δασ96] Δημήτρης Δασκαλάκης. Διαλογική ενημέρωση οντοκεντρικών βάσεων δεδομένων. Master's thesis, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, Φεβρουάριος 1996.

[ABCM92] P. Armenise, S. Bartinelli, C. Chezzi, and A. Morzenti. Software Process Representation Languages: Survey and Assesment. In *4th SEKE*, Capri, Italy, 1992. COMAD.

[AcC92] V. Ambriola, P. Cian carini, and C.Montagero. Software Process Enactment in OIKOS. In *Symposium on softwate development Environments*, Irvine,California, 1992.

[AFN94] J. Kamer A. Finkelstein and B. Nuseibeh, editors. *Software Process Modelling and Technology*. Advanced Software Development Series. Research Study Press, 1994.

[AK94] J.W. Armitage and M. I. Kellner. Conceptual Schema for Process Definitions And Models. In *Proceedings of ICSE94??* IEEE, 1994.

[Boh88] Barry Bohem.
A spiral Model of software developement and Enhancement. *IEEE COMPUTER*, May 1988.

[Boo93] Grady Booch. *Object-oriented Analysis and Design with Applications.* Addison-Wesley Publishing Co, 1993. second edition.

[Bro93] Alan W. Brown. An Examination of the Current State of IPSE Technology. In *Proceedings of the 15th International Conference on Software Engineering*, Berlin, Germany, May 1993.

[BsKH92] I. Ben-shaul, G. Kaiser, and G. Heineman. An architecture for multiuser software development environments. *Software engineering Notes , special Issue*, 17(5), 1992.

[BSW92] B.Peuschel, W. Schaefer, and S. Wolf. A Knowledge based Development Environment (on MERLIN). *International Journal of Software Engineering and Knowledge Engineering*, 2(1):79--106, March 1992.

[Chr94] Alan Christie.
a practical guide to the technology and adoption of software process automation. Technical report, SEI, 1994.

[CJMV95] Panos Constantopoulos, Mathias Jarke, John Mylopoulos, and Yannis Vassiliou.
The Software information Base A Server for Reuse. *VLDB journal*, 1995.

[CKO92] Bill Curtis, Marc I Kellner, and Jim Oliver.
Process Modelling. *COMMUNICATIONS of the ACM*, 35(9), Sep 1992.

[CM88] J. Conklin and M.Bageman. gIBIS : A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, October 1988.

[Cos93] Costas Dadouris, Polykarpos Karamaounas, Nikos Prekas and Maria Theodoridou. SIS Graphical Analysis Interface User's Manual, Working Paper #1. Information Systems and Software Technology Group, Institute of Computer Science Foundation of Research and Technology Hellas, October 1993.

[DBC88] Alan M. Davis, Edward H. Bersoff, and Edward R. Comer.
a strategy for comparing alternative software developement life cycle models. *IEEE Transactions on Software Engineering*, 14(10), October 1988.

[DeM79] T. DeMarco. *Structured Analysis and System Specification.* Yourdon Press, New York, 1979.

[DPS+93] D.Coleman, P.Arnold, S.Bodoff, C. Dollin, H. Cilchrist, F.Hayes, and P. Jeremaes. *Object-oriented Development THE FUDION METHOD* . Prentice-Hall International, 1993.

[ea92] Johnson L. et al. Representation and Presentation of Requirements Knowledge. *IEEE TSE*, 18(10), 1992.

[FKG90] A. Finkelstein, J. Kramer, and M. Goedicke.
viewpoint oriented software developement. In *Proc. of 3rd International Workshop on Software Engineering and its Applications, Toulouse*, Dec 1990.

[G. 94] G. Spanoudakis. *Analogical Similarity of Objects:A Conceptual Modeling Approach.* PhD thesis, Department of Computer Science University of Crete, 1994.

[GR94] Georges GROSZ and Colette ROLLAND.
A General Framework for Describing the Requirements Engineering Process. In *proc of Int'l Conmference on Man Systems and Cybernetics*, San Antonio ,Texas, USA, 1994.

[Har90] D. Harel. Statemate: Aworking environment for the development of complex reactive systems. *IEEE Transactions on software engineering*, 16(4), April 1990.

[HK92] D. Harel and C. Kahana. On staecharts with ovelrapping. *ACM Transactions on Software Engineering and Methodology*, 1(14), Oct 1992.

[IH92] Saimond Ip and Tony Holden.
a knowledge based technique for the process modeling of information systems :the object life cycle diagram. Technical report, ?1992. Caise.

[JC93] M.Letizia Jaccheri and Reidar Conradi. Techniques for Process model Evolution in EPOS. *IEEE ToSE*, 19(12), Dec 1993.

[KFF+90] Mark Kellner, Peter Feiler, Antony Finkelstein, Leon Osterweil, and D. Rombach. SOFTWARE PROCESS EXAMPLE . In *Support for Software Process.* International Software Process Workshop, IEEE -CS Press, October 1990.

[KTO93] William H. Kleppinger, Doris Y. Tamanaha, and Leon J. Osterweil. A Framework for Understanding the Uses of Process Modeling Formalisms. Technical Report UCIrv-93-PROC-ISS-001, 1993.

[Mar92] Martin Doerr, Manos Theodorakis and Polivios Klimathianakis. SIS Data Entry Language Users's Manual, Working Paper #2. Information Systems and Software Technology Group, Institute of Computer Science Foundation of Research and Technology Hellas, December 1992.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Trans. on Inf. Sys.*, 8(4):325, 1990.

[MJC94] Jens-Otto Larsen M.Letizia Jaccheri and Reidar Conradi. Techniques for Process model Evolution in EPOS. In *Proceedings of 4th ICSE* , Capri, Italy, June 1994.

[NFK93] Bashar Nuseibeh, Antony Finkelstein, and Jeff Kramer.
fine grain process modelling. In *7th International Workshop on software Specification and Design (IWSSD-7)*, Redondo Beach,California,USA,6-7DEc, 1993. IEEE CS.

[ROL94] Colette ROLLAND.
A Contextual Approach for the Requirements Engineering Process. In *International Conference on Software Engineering and Knowledge Engineering*, Jurmala ,Latvia, June 1994. 6th.

[SC95] George Spanoudakis and Panos Constantopoulos. Analogical Reuse of Requirements Specifications: A Computational Model. *Automated Arificial Intelligence Journal*, 1995.

[SJ94] Xiping Song and Leon J.Osterweil.
experience with an approach to comparing software design methodologies. *IEEE Transactions on Software Engineering*, 20(5), May 1994.

[Sno95] R. A. Snowdon. Overview of Process Modeling. Technical report, IPDG Manchester Uiversity and ICL Processwise Portfolio Center Kidsgrove, UK, 1995.

[Spa94] G. Spanoudakis. Similarity Analyzer: An Implementation Overview. Technical Report Working Paper No 11, Institute of Computer Science, Foundation of Research and Technology-Hellas, September 1994.

[SSRG96] Samira Si-Said, Colette Rolland, and Georges Grosz. MENTOR : A Computer Aided Requirements Engineering Environment. In *Proceedings of the 8th CAiSE*, May 1996.

Παράρτημα Α

Αγγλοελληνικό Ευρετήριο Όρων

Abstraction	Αφαίρεση
Abstractness	Βαθμός αφαίρεσης
Action	Πράξη
Active Database	Ενεργή βάση δεδομένων
Activity	Δραστηριότητα
Agent	Πράκτορας
Aggregate Object	Σύνθετο αντικείμενο
Aggregation	Συγκρότηση
Analogical Similarity	Αναλογική ομοιότητα
Attribute	Γνώρισμα
Attribution Distance	Απόσταση γνωρίσματος (ολική)
Attribution	Απόδοση γνωρισμάτων
Characticity	Χαρακτηριστικότητα
Choice Context	Περιβάλλον επιλογής
Classification	Ταξινόμηση
Conceptual modeling	Μοντελοποίηση Εννοιών
Configuration Management	Διαχείριση διατάξεων
Context	Περιβάλλον απόφασης
Cooperative Work	Συλλογική εργασία
Data FlowDiagrams	Διαγράμματα ροής δεδομένων
Data Model	Μοντέλο δεδομένων
Decision oriented process model	Αποφασεοκεντρικό μοντέλο διαδικασίας
Description	Περιγραφή
Descriptive process model	Περιγραφικό μοντέλο διαδικασίας
Determinance (of Attribute)	Προσδιοριστικότητα γνωρίσματος
Document	Έγγραφο
Enactment	Ενεργοποίηση , εκτέλεση

Executable formalism	Εκτελέσιμη τυπολογία
Execution Context	Περιβάλλον εκτέλεσης
Formal	Τυπικός
Formalism	Φορμαλισμός , τυπολογία
Generalization	Γενίκευση
Identification distance	Απόσταση ταύτισης
Informal Specification	Άτυπη περιγραφή
Information system	Πληροφοριακό σύστημα
Instance	Περίπτωση
Invariant	Αναλλοίωτος
Methodology	Μεθοδολογία (σύνολο μεθόδων με κοινή προσέγγιση)
Method	Μέθοδος
Modeling Approach	Προσέγγιση στην ανάπτυξη μοντέλων
Module	Τμήμα
Monitor	Παρατηρητής
Notation	Συμβολισμός
Object based	Βασισμένο σε αντικείμενα (οντοκεντρικό)
Object oriented	Αντικειμενοστρεφές
Paradigm	Προσέγγιση
Pattern	Μοτίβο
Plan Context	Περιβάλλον σχεδίου
Prescrpitive process model	Επιτακτικό μοντέλο διαδικασίας
Presentation	Παρουσίαση
Procedural Language	Διαδικαστική γλώσσα
Procedure	Προγραμματιστική διαδικασία
Process Engine	Μηχανισμός εφαρμογής διαδικασίας
Process Modeling	Κατασκευή Μοντέλων Διαδικασιών
Process	Διαδικασία
Product Elements	Στοιχεία προϊόντος
Project Management	Διαχείριση έργου
Project	Έργο
Proscrpitive process model	Προτρεπτικό μοντέλο διαδικασίας
Quality Assurance	Έλεγχος ποιότητας
Repository	Αποθήκη (?)
Representation	Αναπαράσταση
Requirements Analysis	Ανάλυση απαιτήσεων
Requirements Engineering	Τεχνολογία απαιτήσεων λογισμικού
Requirements Specification	Περιγραφή απαιτήσεων

Reverse Engineering	Αναδιάρθρωση
Salience	Σπουδαιότητα
Semiformal	Ημιτυπικός
Similarity Measure	Μέτρο ομοιότητας
Situation Elements	Στοιχεία περίστασης
Situation	Περίσταση
Software Engineering Environment	Περιβάλλον ανάπτυξης λογισμικού
Software System	Σύστημα λογισμικού
Specification	Περιγραφή (τυπική)
State	Κατάσταση
Structured analysis	Δομημένη ανάλυση
Subclass	Υποκλάση
Superclass	Υπερκλάση
Task	Εργασία
Taxonomy	Ταξινόμηση
Trace	Ίχνος
Traceability	Δυνατότητα ιχνηλάτησης
Tracing	Ιχνοληψία
Trigger	Πυροδότηση
Waterfall model	Μοντέλο καταράχτη

Παράρτημα Β

Ελληνοαγγλικό Ευρετήριο Όρων

Αποφασεοκεντρικό μοντέλο διαδικασίας	Decision oriented process model
Περιβάλλον απόφασης	Context
Στοιχεία προϊόντος	Product Elements
Υπερκλάση	Superclass
Άτυπη περιγραφή	Informal Specification
Έγγραφο	Document
Έλεγχος ποιότητας	Quality Assurance
Έργο	Project
Ίχνος	Trace
Ανάλυση απαιτήσεων	Requirements Analysis
Αναδιάρθρωση	Reverse Engineering
Αναλλοίωτος	Invariant
Αναλογική ομοιότητα	Analogical Similarity
Αναπαράσταση	Representation
Αντικειμενοστρεφές	Object oriented
Αποθήκη (?)	Repository
Απόδοση γνωρισμάτων	Attribution
Απόσταση γνωρίσματος (ολική)	Attribution Distance
Απόσταση ταύτισης	Identification distance
Αφαίρεση	Abstraction
Βαθμός αφαίρεσης	Abstractness
Βασισμένο σε αντικείμενα (οντοκεντρικό)	Object based
Γενίκευση	Generalization
Γνώρισμα	Attribute
Διαγράμματα ροής δεδομένων	Data FlowDiagrams
Διαδικασία	Process
Διαδικαστική γλώσσα	Procedural Language

Διαχείριση έργου	Project Management
Διαχείριση διατάξεων	Configuration Management
Δομημένη ανάλυση	Structured analysis
Δραστηριότητα	Activity
Δυνατότητα ιχνηλάτησης	Traceability
Εκτελέσιμη τυπολογία	Executable formalism
Ενεργή βάση δεδομένων	Active Database
Ενεργοποίηση , εκτέλεση	Enactment
Επιτακτικό μοντέλο διαδικασίας	Prescrpitive process model
Εργασία	Task
Ημιτυπικός	Semiformal
Ιχνοληψία	Tracing
Κατάσταση	State
Κατασκευή Μοντέλων Διαδικασιών	Process Modeling
Μέθοδος	Method
Μέτρο ομοιότητας	Similarity Measure
Μεθοδολογία (σύνολο μεθόδων με κοινή προσέγγιση)	Methodology
Μοντέλο δεδομένων	Data Model
Μοντέλο καταράχτη	Waterfall model
Μοντελοποίηση Εννοιών	Conceptual modeling
Μοτίβο	Pattern
Παρατηρητής	Monitor
Παρουσίαση	Presentation
Περίπτωση	Instance
Περίσταση	Situation
Περιβάλλον ανάπτυξης λογισμικού	Software Engineering Environment
Περιβάλλον εκτέλεσης	Execution Context
Περιβάλλον επιλογής	Choice Context
Περιβάλλον σχεδίου	Plan Context
Περιγραφή απαιτήσεων	Requirements Specification
Περιγραφή (τυπική)	Specification
Περιγραφή	Description
Περιγραφικό μοντέλο διαδικασίας	Descriptive process model
Πληροφοριακό σύστημα	Information system
Πράκτορας	Agent
Πράξη	Action
Προγραμματιστική διαδικασία	Procedure
Προσέγγιση στην ανάπτυξη μοντέλων	Modeling Approach

Προσέγγιση	Paradigm
Προσδιοριστικότητα γνωρίσματος	Determinance (of Attribute)
Προτρεπτικό μοντέλο διαδικασίας	Proscrpitive process model
Σπουδαιότητα	Salience
Στοιχεία περίστασης	Situation Elements
Συγκρότηση	Aggregation
Συλλογική εργασία	Cooperative Work
Συμβολισμός	Notation
Σύνθετο αντικείμενο	Aggregate Object
Σύστημα λογισμικού	Software System
Ταξινόμηση	Taxonomy
Ταξινόμηση	Classification
Τεχνολογία απαιτήσεων λογισμικού	Requirements Engineering
Τμήμα	Module
Τυπικός	Formal
Υποκλάση	Subclass
Φορμαλισμός , τυπολογία	Formalism
Χαρακτηριστικότητα	Characticity
Πυροδότηση	Trigger
Μηχανισμός εφαρμογής διαδικασίας	Process Engine

Παράρτημα Γ

Ορισμοί

- Requirements Engineering (RE) : deals with activities which attempt to understand the exact needs of users of a software intensive system and to translate such needs into precise and unambigious statements which will be used in the development of the system - the process of developing a requirements specification has three phases : Elicitation , Specification, Validation

- Method : a disciplined process for generating a set of models that describe various aspects of a software system under development using a well defined notation. Methods instill a discipline into development of complex systems, define products that serve as common vehicles for communication among members of a development team define milestones needed by management to measure progress and manage risk [Booch]

- Methodology : a collection of methods applied across the software development life cycle and unified by some general philosophical approach[Booch]

- Delegation : (εξουσιοδότηση) objects are viewed as prototypes (exemplars) that delegate their behavior to related objects thus eliminating the need for classes

- impedance mismatch : εμφανίζεται μεταξύ των χρηστών ενός συστήματος και αυτών που το αναπτύσσουν , επειδή η κάθε ομάδα στερείται εμπειρίας και ειδικότητας (expertise) στο πεδίο της άλλης.

- process : a set of partially ordered steps intended to reach a goal [Humphray and Feiler SEI-93-TR4]

- process model is an abstract description of an actual or proposed process that represents selected process elements that are important to the purpose of the model and can be enacted by human or machine [CKO92]

- software process :: the activities rules, procedures, techniques and tools used within the business of software production

- object based <> object oriented Object based σημαίνει πως η δομή έιναι βασισμένη στην έννοια του αντικειμένου ενώ object oriented σημαίνει πως τα αντικέιμενα έχουν επιπλέον και μεθόδους που κθορίζουν το πρωτόκολλο αλληλεπίδραασης μεταξύ αντικειμένων.

- Development environments that deal with the development of software based systems ,and integrate process modeling are usualy referd to as IPSEs or PCSEEs **IPSE** (Integrated Project Support Environment) is a computer based system supporting those parts of the overall business process which are concerned with the technical development and maintainance of an information system - the software engineering process.

  It is also called **SDE** ,(Software Development Environment), **SEE** (Software Engineering Environment) and **ISF** (Integrated Software Factory) [Bro93]

- **PCSEE** (Process Centered Software Engineering Environment) Both of them are not well defined terms , in the sense that there is no clear consensus as to what functionality they should support since they have to be general enough to support a great variety of ways of development.

  A lot of research is devoted to the production of integrated process centered software engineering environments. These provide the means to integrate several tools that are not described by process models into life cycle process models.

- **PPL** : process programming languages make explicit the formalisms needed to model and support processes.They may be divided in two classes: modeling and execution(enactment).

- **PML** Process Modeling Language - Γλώσσα παράστασης διαδικασιών

- **Process Program** is a software process description that can be executed on a computer

- **PSE** Process Support Environment (implements PML and tools)

- **PCF** Process Centered Framework (a software product which provides the functionality for the definition and the enactment of a process) it has parts : process enactment /development / debugging services and tool invocation and communication services [Chr94] A tool like ProcessWeaver can be considered a PCF since it has no embedded end-user applications
- **PCE** Process Centered Environment = PCF + Application tools that may support software development , project managemnet , metrics collection etc.
- **PSDE** Process Centered software development Environment
- **software process** total set of sofware engineering activities that are needed to transform users' requirements into software
- In order to reperesent a process , usually the **process elements** that have to be included are : activities and their associated products and dependencies , application tools , human roles and users, organizations and (sub)projects, all guidecd by rules and policies, standards and development methods
- **process engine** is the part of an IPSE that is responsible for enacting the process
- data dictionary (λεξικό δεδομένων) : ένα κεντρικό repository με ορισμούς όρων και εννοιών.

Παράρτημα Δ

Περιγραφές σε Telos